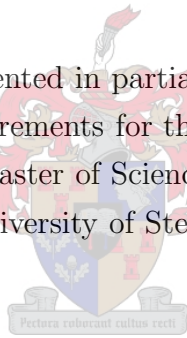


Analytic Models of TCP Performance

Debessay Fesehaye Kassa

Thesis presented in partial fulfilment
of the requirements for the degree of
Master of Science
at the University of Stellenbosch



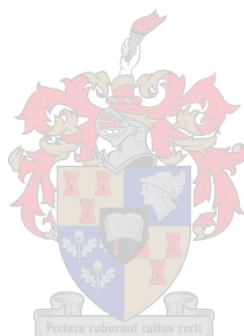
Supervisor: Prof. A. E. Krzesinski

December 2005

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and has not previously in its entirety or in part been submitted at any university for a degree.

Signature



Date

Abstract

The majority of traffic on the Internet uses the Transmission Control Protocol (TCP) as a transport layer protocol for the end-to-end control of information transfer. Measurement, simulation and analytical models are the techniques and tools that can be used to understand and investigate the Internet and its performance. Measurements can only be used to explore existing network scenario or otherwise become costly and inflexible with the growth and complexity of the Internet. Simulation models do not scale with the growth of network capacities and the number of users. Computationally efficient analytical models are therefore important tools for investigating, designing, dimensioning and planning IP (Internet Protocol) networks.

Existing analytical models of TCP performance are either too simple to capture the internal dynamics of TCP or are too complex to be used to analyze realistic network topologies with several bottleneck links. The literature shows that the fixed point algorithm (FPA) is a very useful way of solving analytical models of Internet performance. This thesis presents fast and accurate analytical models of TCP performance with the FPA used to solve them.

Apart from what is observed in experimental literature, no comprehensive proof of the convergence and uniqueness of the FPA is given. In this thesis we show how the FPA of analytical models of reliable Internet protocols such as TCP converges to a unique fixed point. The thesis specifies the conditions necessary in order to use the FPA for solving analytical models of reliable Internet protocols. We also develop a general implementation algorithm of the FPA of analytical models of TCP performance for realistic and arbitrary network topologies involving heterogenous TCP connections crossing many bottleneck links.

The models presented in this thesis give Internet performance metrics, assuming that only basic network parameters such as the network topology, the number of TCP connections, link capacity, distance between network nodes and router buffer sizes are known. To obtain the performance metrics, TCP and network sub-models are used. A closed network of G/∞ queues is used to develop each TCP sub-model where each queue represents a state of a TCP connection. An $M/M/1/K$ queue is used for each network sub-model which represents the output interface of an IP router with a buffer capacity of $K - 1$ packets. The two sub-models are iteratively solved.

We also give closed form expressions for important TCP performance values and distributions. We show how the geometric, bounded geometric and truncated geometric distributions can be used to model reliable protocols such as TCP. We give models of the congestion window *cwnd* size distribution by conditioning on the slow start threshold *ssthresh* distribution and vice-versa. We also present models of the probabilities of TCP timeout and triple duplicate ACK receptions.

Numerical results based on comparisons against *ns2* simulations show that our models are more accurate, simpler and computationally more efficient than another well known TCP model. Our models can therefore be used to rapidly analyze network topologies with several bottlenecks and obtain detailed performance metrics.



Opsomming

Die meerderheid van die verkeer op die Internet gebruik die Transmission Control Protocol (TCP) as 'n vervoer laag protokol vir die einde-tot-einde kontrole van inligting oordrag. Meting, simulاسie en analitiese modelle is die tegnieke en gereedskap wat gebruik kan word om die Internet te ondersoek en verstaan. Meting kan slegs gebruik word om bestaande netwerke scenarios te verken. Meting is duur en onbuigsaam met die groei en samegesteldheid van die Internet. Simulasie modelle skaal nie met die groei van netwerk kapasiteit en gebruikers nie. Analitiese modelle wat berekening effektief is is dus nodige gereedskap vir die ondersoek, ontwerp, afmeting en beplanning van IP (Internet Protocol) netwerke.

Bestaande analitiese TCP modelle is of te eenvoudig om die interne dinamiek van die TCP saam te vat of hulle is te ingewikkeld om realistiese netwerk topologie met heelwat bottelnek skakels te analiseer. Literatuur toon dat die fixed point algorithm (FPA) baie handig is vir die oplos van analitiese modelle van Internet verrigting. In hierdie tesis word vinnige en akkurate analitiese modelle van TCP verrigting opgelos deur FPA weergegee.

Buiten wat deur eksperimentele literatuur aangedui word is daar geen omvattende bewyse van die konvergensie en uniekheid van die FPA nie. In hierdie tesis word aangedui hoe die FPA van analitiese modelle van betroubare Internet protokolle soos die TCP konvergeer na 'n unieke vaste punt. Hierdie tesis spesifiseer die voorwaardes benodig om die FPA te gebruik vir die oplos van analitiese modelle van realistiese Internet protokolle. 'n Algemene uitvoer algoritme van die FPA van analitiese modelle van TCP vir realistiese en arbitrêre netwerk topografie insluitende heterogene TCP konneksies oor baie bottelnek skakels is ontwikkel.

Die model in hierdie tesis gee Internet verrigting metodes met die aanname dat slegs basiese netwerk parameters soos netwerk topologie, die aantal TCP konneksies, die konneksie kapasiteit, afstand tussen netwerk nodusse en die roete buffer groter bekend is. Om die verrigting metodes te verkry, word TCP en netwerk sub-modelle gebruik. 'n Geslote netwerk van $/G/\infty$ ry is gebruik om elke TCP sub-model, waar elke ry 'n toestand van 'n TCP konneksie voorstel, te ontwikkel. 'n $M/M/1/K$ ry is gebruik vir elke netwerk sub-model wat die uitset koppelvlak van 'n IP roetemaker met 'n buffer kapasiteit van $K - 1$ pakkies voorstel. Die twee submodelle word iteratief opgelos.

Geslote vorm uitdrukkings vir belangrike TCP verrigting waardes en verspreidings word gegee. Daar word getoon hoe geometriese, begrensde geometriese en geknotte geometriese verspreidings gebruik kan word om betroubare protokolle soos die TCP te modelleer. Modelle van die kongestie venster *cwnd* grootte verspreiding word gegee deur die kondisionering van die

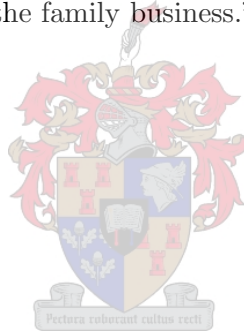
stadige aanvang drempel *ssthresh* verspreiding en andersom. Modelle van die voorspelling van TCP tyduit en trippel duplikaat ACK resepsie word weergegee.

Numeriese resultate gebaseer op vergelykings met *ns2* simulasies wys dat ons modelle meer akkuraat, eenvoudiger en berekeningsgewys meer effektief is as ander wel bekende TCP modelle. Ons modelle kan dus gebruik word vir vinnig analyse van netwerk topologie met verskeie bottelnekke en om gedetailleerde verrigting metodes te bekom.



Dedication

I dedicate this work to my father **Fesehaye Kassa Weldemichael**, my mother **Leteberhan Beyene Menkerios**, and my eldest brother **Yemane Fesehaye Kassa** for all the ups and downs they endured to see my success, fruition and happy life. They said, “Just study, we will take care of you and the rest of the family business.”



Acknowledgements

I thank the Eritrean government for the full sponsorship of my honours degree, and most of this work through its *Human Resource Development (EHRD)* program. I greatly value the financial support of TELKOM SA and SIEMENS for the rest of this work through the University of Stellenbosch unit of the *Telkom-Siemens Centre of Excellence in ATM & Broadband Networks & their Applications*.

I also thank each person who in some way has contributed to the success of this work.



List of publications

1. Debessay Fesehaye and A. E. Krzesinski. A fast and accurate analytical model of TCP performance. In Proceedings of the Southern African Telecommunication Networks and Applications Conference (SATNAC 2005), Central Drakensberg, KwaZulu Natal, South Africa, September 2005.
2. Debessay Fesehaye and A. E. Krzesinski. A queueing network model of TCP performance. In Proceedings of the South African Institute for Computer Scientists and Information Technologists (SAICSIT 2005), to appear in ACM International Conference Proceedings Series, White River, South Africa, September 2005.



Contents

1	Introduction	1
1.1	Problem statement	2
1.1.1	Network topology	3
1.1.2	Input and output variables	4
1.2	Structure of the Thesis	6
2	An overview of TCP/IP	7
2.1	The Internet protocol suit	7
2.1.1	The link layer	7
2.1.2	The network layer	8
2.1.3	The transport layer	9
2.1.4	The application layer	9
2.2	TCP services and basics	10
2.3	Connection establishment	11
2.4	Implementations of TCP	11
2.5	TCP–Tahoe	12
2.5.1	Slow Start	12
2.5.2	Packet losses and Fast Retransmit	13

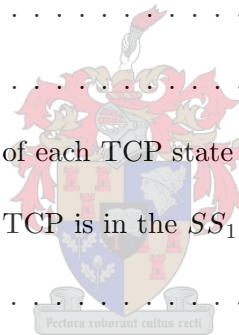
2.5.3	Congestion Avoidance	14
2.6	Round trip time measurement and timeout calculation	15
2.6.1	The exponential back-off	16
2.6.2	Karn's algorithm	17
2.7	The TCP sliding window algorithm	17
2.8	Queue management in TCP/IP networks	18
2.9	Statistical multiplexing	18
2.10	Some basic definitions	19
2.10.1	Throughput	19
2.10.2	Bandwidth–delay product	19
2.11	MPLS networks	19
3	A short survey of analytical models of TCP	21
3.1	Classification of analytical models of TCP	21
3.1.1	The RTT and the packet loss probability are known or unknown . . .	21
3.1.2	The flows are finite or infinite	23
3.1.3	The mathematical technique used	23
3.2	Our modeling technique	26
3.3	Chapter summary	28
4	Mathematical background	29
4.1	The fixed point algorithm	29
4.2	The simulation model	30
4.2.1	The batch means technique of simulation	30
4.3	Some basic concepts of probability	31

4.4	Probability distributions	32
4.4.1	Binomial distribution	32
4.4.2	Geometric distribution	32
4.5	Stochastic processes	33
4.6	Queueing theory	33
4.6.1	The $M/M/1/K$ system	34
4.6.2	The $.G/\infty$ system	34
4.7	Queueing networks	34
4.7.1	The mean value algorithm (MVA): A single class closed queueing network of $.G/\infty$ queues	35
4.8	The hazard rate function	36
4.9	Chapter summary	36
5	Convergence of the FPA used to study reliable Internet protocols (TCP)	37
5.1	Brouwer's fixed point theorem	38
5.2	FPA of analytical models of TCP performance	39
5.3	The two dimensional FPA for a single bottleneck link	39
5.3.1	The compact and convex set (region) where the FPA is carried out	39
5.3.2	The continuous fixed point function	40
5.3.3	Uniqueness of the fixed point of TCP models	43
5.4	A multi dimensional FPA for a single bottleneck link	46
5.5	The FPA of analytical models of TCP for a multi bottleneck network	47
5.6	Chapter summary	50
6	The six-state analytical models of TCP	52

6.1	The TCP sub-model	53
6.1.1	The states of TCP connections	54
6.1.2	The transition probability matrix	55
6.1.3	The equilibrium probabilities	63
6.1.4	The effects of exponential back-off	65
6.1.5	The service times	66
6.1.6	Mean value analysis	67
6.1.7	The load offered to the network sub-model	67
6.2	The network sub-model	68
6.2.1	The average buffer occupancy	69
6.2.2	The average round trip time	72
6.3	The multi bottleneck scenario	73
6.4	Individual TCP connections	74
6.5	The complexity of the models	74
6.6	Chapter summary	75
7	Numerical Results: Single bottleneck link analysis	77
7.1	The validation of the models	77
7.2	The topology of the Internet-A European network	78
7.3	The loss probability	79
7.4	Average <i>cwnd</i> and <i>ssthresh</i>	80
7.5	The <i>cwnd</i> size distribution	82
7.6	The <i>ssthresh</i> distribution	90
7.7	The timeout probability	101

7.8	The buffer occupancy	104
7.9	Analysis of Individual Connections	106
7.10	Chapter summary	106
8	Numerical Results: Multi bottleneck link analysis	112
8.1	The analysis of multi bottleneck links	112
8.1.1	Implementation details of the multi bottleneck topology	114
8.1.2	A general implementation algorithm of the FPA for multi bottleneck links	114
8.2	The loss probability on the multi bottleneck links	115
8.3	The average <i>cwnd</i> on the multi bottleneck links	121
8.4	An analysis of fat pipes	121
8.5	Chapter summary	121
9	Thesis summary and work in progress	132
9.1	Thesis summary	132
9.2	Work in progress	134
A	The derivation of P_{L_f}	135
B	State transitions	137
B.1	The detailed state transition graph	137
B.2	The idealized TCP transmission cycle	139
B.3	The $P(A_i, B_j)$'s and their derivation	141
B.3.1	Timeout and fast retransmit losses	142
B.3.2	The slow start threshold	142
B.3.3	The transition probabilities and their derivation	143

B.4	Chapter summary	151
C	The service times of each TCP queue	152
D	The number of packets offered by each TCP queue	155
E	More models of TCP	160
E.1	The TCP <i>cwnd</i> size distribution	160
E.1.1	The <i>cwnd</i> ₁ model	161
E.1.2	The <i>cwnd</i> ₂ model	162
E.2	The TCP timeout and fast-retransmit probabilities	162
E.3	The <i>ssthresh</i> models	163
E.3.1	The <i>ssthresh</i> ₁ model	163
E.3.2	The <i>ssthresh</i> ₂ model	163
E.4	The stationary probabilities of each TCP state	164
E.4.1	The probability that TCP is in the <i>SS</i> ₁	165
E.5	Chapter summary	166
F	An Eritrean network topology	168



List of Tables

1.1	The input variables used in and the performance metrics obtained from the models presented in this thesis	5
3.1	Advantages and disadvantages of renewal theory models of TCP	24
3.2	Advantages and disadvantages of fixed point models of TCP	24
3.3	Advantages and disadvantages of control theoretic models of TCP	25
3.4	Advantages and disadvantages of processor sharing models of TCP	26
3.5	Advantages and disadvantages of fluid models of TCP	27
6.1	The maximum and minimum <i>cwnd</i> values in each state	56
6.2	The four analytical models	62
6.3	The transition probability matrix	63
7.1	Single bottleneck loss probability comparison of the models	80
7.2	Comparison of the <i>cwnd</i> size distributions	91
7.3	Buffer occupancy comparison of the $M/M/1/K$ and geometric buffer formulas	104

List of Figures

1.1	The network scenario	4
2.1	The four layers of the TCP/IP protocol suite	8
2.2	Encapsulation of TCP data in an IP datagram	10
2.3	TCP-Tahoe <i>cwnd</i> evolution	15
2.4	Visualization of TCP sliding window	17
3.1	The iterative solution procedure with the interaction between the TCP and the network sub-models	22
6.1	The iterative solution procedure	52
6.2	The state transition diagram of greedy TCP-Tahoe	54
6.3	The buffer occupation of TCP for a fixed number of concurrent TCP connections	70
6.4	The window evolution for a TCP connection	71
7.1	The network topology	78
7.2	The packet loss probability when a TCP tick value of 500ms and the $M/M/1/K$ buffer formula are used	81
7.3	The packet loss probability (log-scale) when the TCP tick value is 500ms, the buffer capacity is 128 packets and the $M/M/1/K$ buffer formula is used . . .	82

7.4	The packet loss probability when a TCP tick value of 100ms and the $M/M/1/K$ buffer formula are used	83
7.5	The packet loss probability when a TCP tick value of 500ms and geometric buffer formula are used	84
7.6	The packet loss probability when a TCP tick value of 100ms and geometric buffer formula are used	85
7.7	Average <i>cwnd</i> size and <i>ssthresh</i> when a TCP tick value of 500ms and the $M/M/1/K$ buffer formula are used	86
7.8	Average <i>cwnd</i> size and <i>ssthresh</i> when a TCP tick value of 100ms and the $M/M/1/K$ buffer formula are used	87
7.9	Average <i>cwnd</i> size and <i>ssthresh</i> when a TCP tick value of 500ms and geometric buffer formula are used	88
7.10	Average <i>cwnd</i> size when a TCP tick value of 100ms and geometric buffer formula are used	89
7.11	The <i>cwnd</i> size distribution for 25 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms	92
7.12	The <i>cwnd</i> size distribution for 25 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms	93
7.13	smoothed and non-smoothed <i>cwnd</i> size distributions for 25 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms	94
7.14	The <i>cwnd</i> size distribution when the buffer capacity is 128 packets and the TCP tick value is 500 ms for 100 active TCP connections	95
7.15	The <i>cwnd</i> size distribution when the buffer capacity is 128 packets and the TCP tick value is 500 ms for 400 active TCP connections	96
7.16	The <i>ssthresh</i> distribution for 25 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms	97
7.17	The <i>ssthresh</i> distribution for 25 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms	98

7.18	The <i>ssthresh</i> distribution for 100 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms	99
7.19	The <i>ssthresh</i> distribution for 400 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms	100
7.20	Fraction of time TCP waits for timeout to expire when the buffer capacity is 128 packets	102
7.21	Fraction of time TCP waits for timeout to expire when the buffer capacity is 512 packets	103
7.22	Comparisons of some performance metrics which are not easy to obtain using <i>ns2</i> simulations	105
7.23	Buffer occupancy with a TCP tick value of 100 ms	107
7.24	Buffer occupancy with the standard TCP tick value of 500 ms	108
7.25	Throughput of individual connections when the buffer capacity is 512 packets	109
7.26	Throughput of individual connections: the buffer capacity is 128 packets, the TCP tick value is 100 ms (0.1s) and 500 ms (0.5s) when the number of active TCP connections is 200	110
8.1	The two bottleneck link case: mapping the network topology in Figure 7.1 into sub-models	113
8.2	An example of the input file	115
8.3	Packet loss probability on the multi bottleneck links when the TCP tick value is 500ms and buffer size is 128 packets	117
8.4	Packet loss probability on the multi bottleneck links when the TCP tick value is 500ms and buffer size is 128 packets	118
8.5	Packet loss probability on the multi bottleneck links when the TCP tick value is 100ms and buffer size is 128 packets	119
8.6	Packet loss probability on the multi bottleneck links when the TCP tick value is 100ms and buffer size is 128 packets	120

8.7	Packet loss probability on the multi bottleneck links when a TCP tick value of 500 ms and buffer size of 128 packets are used	122
8.8	Packet loss probability on the multi bottleneck links when a TCP tick value of 100 ms and buffer size of 128 packets are used	123
8.9	Average <i>cwnd</i> on the multi bottleneck links when the TCP tick value is 500ms and the buffer size is 128 packets	124
8.10	Average <i>cwnd</i> on the multi bottleneck links when the TCP tick value is 500ms and the buffer size is 128 packets	125
8.11	Average <i>cwnd</i> on the multi bottleneck links when the TCP tick value is 100ms and the buffer size is 128 packets	126
8.12	Average <i>cwnd</i> on the multi bottleneck links when the TCP tick value is 100ms and the buffer size is 128 packets	127
8.13	Average <i>cwnd</i> : the buffer capacity is 128 packets and the TCP tick value is 500 ms	128
8.14	Average <i>cwnd</i> on the multi bottleneck links: the buffer capacity of 128 packets and the TCP tick value is 100 ms	129
8.15	Multi bottleneck performance metrics when a TCP tick value of 500 ms and a buffer capacity of 512 packets are used	130
B.1	A detailed description of the transitions of the TCP model where the exponential back-off is built into the <i>SS</i> and <i>SST</i> states of our models	138
B.2	A description of the transitions of the TCP model when the exponential back-off is separately shown.	140
B.3	The idealized TCP cycle when <i>ssthresh</i> = 6	141
B.4	The sliding window protocol when <i>ssthresh</i> = 4	145
F.1	The path followed by Internet connections from the MOE clients to servers in the USA	168

Chapter 1

Introduction

Communication networks and the Internet in particular have shown tremendous growth. This growth in the use and capabilities of communication networks has transformed the way we live and work. As society progresses further into the information age, the reliance on networking will increase. The explosive growth of data traffic is accompanied by rapid change and great heterogeneity of the communication links, the protocols that interact over the links and the various applications used with the protocols.

Measurement, simulation and analytical models are the techniques and tools that can be used to understand and investigate the Internet and its performance. Measurements can only be used to explore existing network scenario or otherwise become costly and inflexible with the growth and complexity of the Internet. Simulation models do not scale with the growth of network capacities and the number of users. Computationally efficient analytical models are therefore important tools for investigating, designing, dimensioning and planning IP (Internet Protocol) networks.

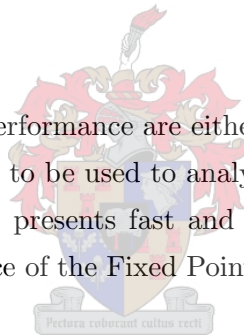
The majority of traffic on the Internet uses the Transmission Control Protocol (TCP) as a transport protocol. According to measurements [88, 89] performed on a commercial backbone, about 95% of the total bytes and 90% of the total packets are transmitted by TCP. The TCP protocol plays a key role in delivering a reliable service to widely used network applications such as e-mail programs, web browsers and FTP file downloads. Developing, implementing and analyzing accurate and fast analytical models for TCP behavior is therefore necessary for predicting the quality of service (QoS) and for the design, planning, configuration, management and dimensioning of IP networks. The availability of these tools allows operators of both telecom and computer networking areas (carriers, companies, Internet Service Providers) to better design their networks, with remarkable advantages in terms of cost reductions and

better user-perceived QoS.

Network dimensioning [91] is responsible for mapping traffic requirements to the physical network resources and for providing provisioning directives in order to accommodate the predicted traffic demands. So far there are no satisfactory solutions to the problem of dimensioning IP networks. Many network operators and Internet Service Providers (ISPs) often resort to overprovisioning in order to comply with the Service Level Agreements (SLAs) established with their clients when dimensioning their networks. Current approaches often rely on Poisson assumptions for the user-generated packet flows, thus neglecting the recent findings on the presence of correlations in network traffic, as well as neglecting the closed-loop congestion control algorithms implemented at the transport layer. Hence estimating the quality of service and dimensioning IP networks should be based on fast models capable of accurately estimating the Internet performance metrics. In the following sections we present the problem statement and the structure of this thesis.

1.1 Problem statement

Existing analytical models of TCP performance are either too simple to capture the internal dynamics of TCP or are too complex to be used to analyze realistic network topologies with several bottleneck links. This thesis presents fast and accurate analytical models of TCP performance with proof of convergence of the Fixed Point Algorithm (FPA) used to solve the analytical models.



The first problem addressed in this thesis is how to show that the FPA of analytical models of TCP converges to a unique fixed point.

The second problem addressed in this thesis is how to compute TCP performance metrics such as those described as output variables in section 1.1.2 below. Our models are faster and more accurate than several well known models. The basic network parameters for our models are described as input parameters in section 1.1.2 for a network topology such as the one described in section 1.1.1.

The third problem addressed is how the geometric, bounded geometric and truncated geometric distributions can be used to model reliable protocols such as TCP.

The fourth problem addressed is how to develop a general implementation algorithm of the FPA of analytical models of TCP performance for arbitrary and realistic network topologies involving heterogeneous TCP connections crossing many bottleneck links.

The last problem addressed in this thesis is how the congestion window *cwnd* size distribution can be calculated by conditioning on the slow start threshold *ssthresh* distribution and vice-versa and how the probabilities of TCP timeout and triple duplicate ACK receptions can be obtained using closed form expressions.

1.1.1 Network topology

The models presented in this thesis are used to analyze network topologies such as the one given in Figure 1.1. The IP backbone (the network core) of the topology is described by a directed graph $G = (V, E)$ where the nodes (vertices) represent the routers, and the edges represent the (unidirectional) links connecting pairs of nodes within the network backbone. Routers forward packets in the network according to a routing strategy that is assumed to be fixed and known a priori. In multi protocol label switching (MPLS) networks (see section 2.11) a single label switching router (LSR), generally the ingress or egress LSR, specifies some or all of the LSRs in an *explicit routed* label switching path (LSP) which is set up before forwarding of packets can commence. Routers at the edge of the network are called *edge* routers or *boundary* routers and the interior routers within the network backbone are called *core* routers.

Packets waiting to be transmitted over a link are queued at routers according to a FIFO queueing discipline. We assume that router interfaces are equipped with output buffers where packets having the same next hop are queued. This is how routers are represented in the network simulator *ns2*. In real TCP connections data packets are exchanged in both directions, however the bulk of the data is usually sent in one direction, especially when large files are transferred between hosts. For this reason, the use of unidirectional TCP connections, which simplifies the description of the traffic, is commonly accepted in the literature.

In this thesis the focus is on greedy or long lived TCP connections which always have data to send. The number of greedy flows does not change over time. After an initial transient corresponding to the start-up of the flows, the system reaches an equilibrium point that is a steady state for both the network and the individual flows. Analytical models of greedy flows usually disregard the initial transient phase, assuming that the network quickly operates at the steady state. Hence when greedy flows are simulated an initial transient period is discarded in the measurement collection process.

Figure 1.1 shows a number of small clouds attached to the main cloud. The small clouds represent local area networks (LANs) or other portions of the global network and the main

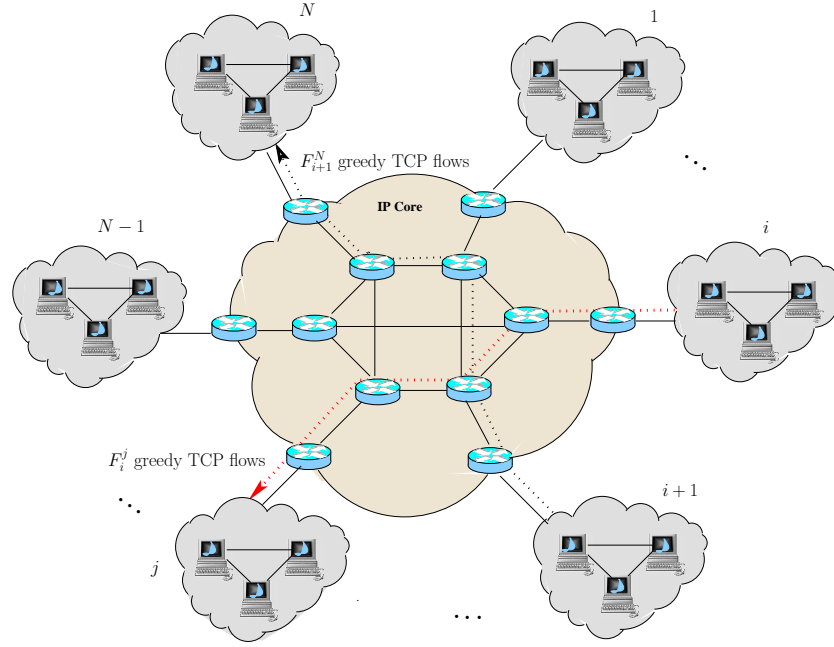


Figure 1.1: The network scenario

cloud corresponds to the IP core. Consider (greedy) TCP flows F_i^j from source i to destination j as shown in the figure which congest the network. Each group of such flows (flows with the same path, TCP version, the same packet size ...) is represented by a *TCP sub-model*. Each forwarding equivalence class (FEC) of TCP connections of an MPLS network (see section 2.11) can be treated as a TCP sub-model. Each TCP connection suffers packet loss and delay in one or more of the bottleneck links it crosses. Each of these links with the respective router represents a *network sub-model*.

The TCP connections send data to the network and the network controls the rate at which data is sent by the TCP connections by dropping packets to signal congestion. This feed-back procedure is modeled with a Fixed Point Algorithm (FPA). The FPA, the TCP and network sub models are explained and analyzed in subsequent chapters.

This thesis presents models of greedy TCP connections which can be extended so as to build a comprehensive model of the Internet.

1.1.2 Input and output variables

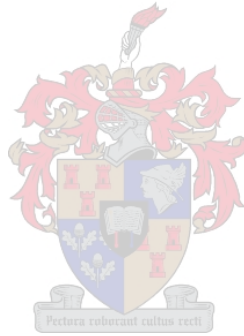
The input variables and the performance metrics which are the output variables obtained from the models presented in this thesis are given in Table 1.1.

Input parameters	<p>The network topology represented by the graph G,</p> <p>the (constant) capacity and propagation delay of each link,</p> <p>the (arbitrarily distributed) propagation delays suffered by the packets arriving at each edge router from outside the network core,</p> <p>the capacity of the external links attached to each edge router,</p> <p>the (fixed) routing (path) of packets within the network,</p> <p>the router buffer capacity (in packets) associated with each link,</p> <p>the number of TCP flows established between each pair of edge routers,</p> <p>the size of data packets sent by the TCP sources and the size of the ACK packets sent by the receivers, and</p> <p>the TCP version (implementation) and the maximum window size.</p>
Performance metrics	<p>The traffic intensity ρ defined by the product of the average arrival rate of packets at the router queue and the average transmission time of packets over the link,</p> <p>the average packet loss probability P_L,</p> <p>the average link utilization u, which is equal to $\rho(1 - P_L)$,</p> <p>the average queue length and the average time spent by a packet at a queue,</p> <p>the average Round Trip Time (RTT),</p> <p>the <i>cwnd</i> size and <i>ssthresh</i> distributions of the TCP sources and</p> <p>the average throughput which is the arrival rate of packets at the receivers.</p>

Table 1.1: The input variables used in and the performance metrics obtained from the models presented in this thesis

1.2 Structure of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents an overview of TCP/IP. Chapter 3 gives a short survey of analytical models of TCP. Chapter 4 explains some mathematical background of the TCP models presented in this thesis. Chapter 5 shows the convergence of the FPA used to study reliable Internet protocols (TCP). Chapter 6 presents several six state analytical models of TCP. Chapters 7 and 8 present numerical results of a single and multi bottleneck topologies respectively. Finally Chapter 9 presents the summary of the thesis and on going work.



Chapter 2

An overview of TCP/IP

This chapter presents an overview of the mechanisms of the Transmission Control Protocol (TCP) which are relevant to the TCP models presented in this thesis. It is not meant to give a detailed description of the TCP protocol. We present the Internet (TCP/IP) protocol suit, the TCP services and basics, the Tahoe implementation of TCP, the round trip time measurement and timeout calculation, the TCP sliding window algorithm, queue management in TCP/IP networks, the statistical multiplexing, some basic definitions and MPLS networks in sections 2.1 through 2.11 respectively. The discussion in this chapter follows the specifications of TCP/IP given in [26, 54, 76, 83].

2.1 The Internet protocol suit

The Internet architecture, which is sometimes called the TCP/IP architecture after its two main protocols evolved from an earlier packet-switched network called the ARPANET. Both the Internet and the ARPANET were funded by the Advanced Research Projects Agency (ARPA), one of the R & D funding agencies of the U.S.A. Department of Defense.

The TCP/IP protocol suite otherwise called the Internet protocol suit allows computers from different vendors with different architectures and running different operating systems to communicate with each other. It is a combination of protocols at four layers as shown in figure 2.1.

2.1.1 The link layer

The *link* layer, sometimes called the *data-link* layer or *network interface* layer, normally includes the network interface card (NIC) driver in the operating system and the corresponding

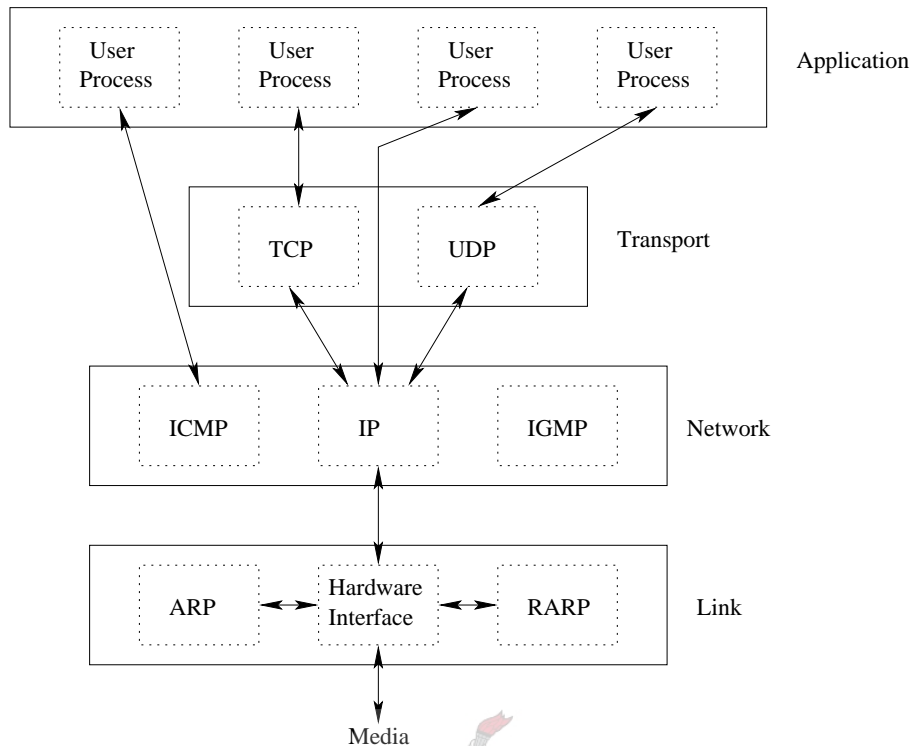


Figure 2.1: The four layers of the TCP/IP protocol suite

NIC in the computer. Together they handle all the hardware details of physically interfacing with the cable (or whatever type of media is being used).

The ARP (Address Resolution Protocol) and RARP (Reverse Address Resolution Protocol) are specialized protocols used only with certain types of network interfaces (such as Ethernet and token ring) to convert between the addresses used by the IP layer and the addresses used by the NIC. For more on these protocols see Chapters 4 and 5 of [83].

2.1.2 The network layer

The *network* layer (sometimes called the *internet* layer) handles the movement of packets around the network. The routing of packets, for example, takes place here. The IP (Internet Protocol), ICMP (Internet Control Message Protocol), and IGMP (Internet Group Management Protocol) provide the network layer in the TCP/IP protocol suite.

The IP protocol in the network layer is used by both TCP and UDP (User Datagram Protocol). All TCP and UDP data that is transferred through an internet goes through the IP layer at both end systems and at every intermediate router. Figure 2.1 also shows an appli-

cation accessing IP directly. This is rare, but possible. Some older routing protocols were implemented this way. Also, it is possible to experiment with new transport layer protocols using this feature.

The ICMP (Internet Control Message Protocol) is an adjunct to IP. It is used by the IP layer to exchange error messages and other control information with the IP layer in another host or router. For more on this protocol see Chapter 6 of [83]. Although ICMP is used primarily by IP, it is possible for an application to access it. The two popular diagnostic tools, ping and traceroute (see Chapters 7 and 8 of [83]), both use ICMP.

IGMP is the Internet Group Management Protocol. It is used when multicasting to send a UDP datagram to multiple hosts. For more on multicasting and IGMP see chapters 12 and 13 of [83].

2.1.3 The transport layer

The *transport* layer provides a flow of data between two hosts, for the application layer above it. The TCP/IP protocol suite offers two transport protocols: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

TCP provides a reliable flow of data between two hosts. It is concerned with issues such as dividing the data passed to it from the application into appropriately sized segments for the network layer below, acknowledging received packets, setting timeouts to make certain the destination acknowledges packets that are sent, and so on. Because this reliable flow of data is provided by the transport layer, the application layer can ignore these details.

UDP on the other hand provides a simpler service to the application layer. It sends packets of data called datagrams from one host to the other, but there is no guarantee that the datagrams reach their destination. Reliability must be added by the application layer.

2.1.4 The application layer

The *application* layer handles the details of the particular application. Applications are normally user processes. There are many common TCP/IP applications that almost every implementation provides. For instance TCP is used by many of the popular applications, such as telnet, RLogin (Remote Login), FTP (File Transfer Protocol) and SMTP (Simple Mail Transfer Protocol or e-mail). The Domain Name System (DNS), the Trivial File Transfer Protocol (TFTP), the Simple Network Management Protocol (SNMP) and the Bootstrap Protocol (for

diskless systems) use UDP applications. For details about these different protocols see the respective chapters of [83].

The complex TCP protocol plays a large role in the Internet. In the following sections we discuss some details of TCP.

2.2 TCP services and basics

Even though TCP and UDP use the same network layer (IP), TCP provides a different service to the application layer than UDP does. TCP provides a connection-oriented, reliable, byte stream service.

The term *connection-oriented* means that the two applications using TCP (normally considered a client and a server) must establish a TCP connection with each other before they can exchange data.

TCP segments are transmitted in IP datagrams as shown in figure 2.2. IP datagrams can be duplicated and arrive out of order. Hence TCP segments can also be duplicated and arrive out of order. To achieve *reliability* TCP packetizes the user data into segments, sets a timeout when it sends data, acknowledges data received by the destination, reorders out-of-order data, discards duplicate data, provides end-to-end flow control, and calculates and verifies a mandatory end-to-end checksum.

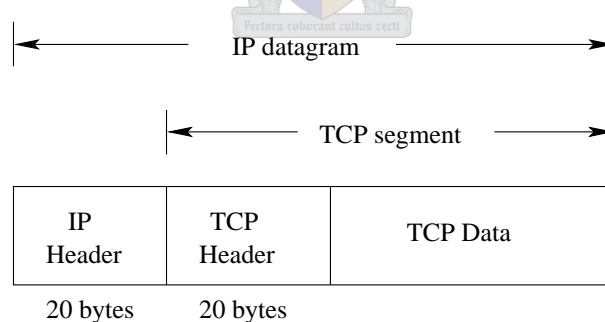


Figure 2.2: Encapsulation of TCP data in an IP datagram


A stream of bytes is exchanged across the TCP connection between the two applications. No record markers are inserted by TCP. This is called a *byte stream service*.

2.3 Connection establishment

Before two processes can exchange data using TCP, they must establish a connection between themselves. When two processes have completed their data transfer they terminate the connection. A description of how connections are established using a three-way handshake and terminated is given in chapter 18 of [83].

During connection establishment, each end of the connection advertises a maximum segment size (MSS) which is the maximum number of bytes allowed in the data payload of the TCP packet. For many BSD (Berkeley Software Distribution) implementations, the MSS must be a multiple of 512 bytes, and the default MSS advertised is 1024 bytes. For other systems such as SunOS 4.1.3, Solaris 2.2 and AIX 3.2.2, the default MSS advertised is 1460 bytes. For these systems, the MSS is set to a 536 bytes and for many BSD implementations it is set to 512 bytes if either the two ends of the connection are not on the same local Ethernet, or if one end does not receive an MSS indication. This 536 MSS value allows for a 20-byte IP header and a 20-byte TCP header to fit into a 576-byte IP datagram.

2.4 Implementations of TCP



The de facto standard for TCP/IP implementations was developed by the Computer Systems Research Group at the University of California at Berkeley. Historically this has been distributed with the 4.x BSD (Berkeley Software Distribution) system, and with the “BSD Networking Releases”. This source code has been the starting point for many other implementations. The 4.3 BSD Tahoe is one of the most commonly used implementations of TCP from which most other implementations are derived. Among the other implementations of TCP are Reno [15], NewReno [33], Lite [83], SACK (Selective Acknowledgement) [65], FACK (Forward Acknowledgement) [64] and Vegas [18, 19].

There have been several studies comparing the performance of these different implementations of the TCP protocol. The better performance of TCP-Tahoe when compared with the other implementations is shown in [68]. Sikdar *et al.* [82] have also shown that as losses become correlated, Tahoe can outperform both the Reno and SACK implementations of TCP. Besides as stipulated in [68, 29] the Reno, New Reno and SACK implementations of TCP which are derived from Tahoe and the TCP-Vegas have some performance problems and there is an ongoing effort to improve these implementations of TCP.

In the next section we present the Tahoe implementation of TCP.

2.5 TCP–Tahoe

TCP–Tahoe is the original implementation of Van Jacobson’s proposed mechanisms [45]. It is a follow up of the original TCP that was standardized in RFC 793 in September 1981 and is one of the most widely used implementations of the TCP protocol. The Tahoe TCP implementation added a number of new algorithms and refinements to earlier implementations. The new algorithms include *Slow–Start*, *Congestion Avoidance*, and *Fast Retransmit*. We next describe these algorithms as explained in RFC 2001 and [83].

2.5.1 Slow Start

The slow start (*SS*) algorithm solves the problems of previous TCP implementations which start a connection with the sender injecting multiple segments into the network, up to the window size advertised by the receiver. The approach used by the previous implementations drastically reduces the throughput of a TCP connection as shown in [45] if there are routers and slower links between the sender and the receiver. This is because some intermediate routers must queue the packets, and the routers can run out of buffer space.

In *SS* TCP uses a window called congestion window (*cwnd*) to restrict data flow to less than the receiver’s buffer size when congestion occurs. Hence at any time, TCP acts as if the window size is:

$$Allowed_window = \min(advertised_window, congestion_window)$$

where the *congestion window* is flow control imposed by the sender and the *advertised window* is flow control imposed by the receiver. The former is based on the sender’s assessment of perceived network congestion; the latter is related to the amount of available buffer space at the receiver for the connection. As explained in section 13.20 of [26] in steady state on a non–congested connection the *cwnd* is the same as the receiver’s window.

When a new connection is established with a host on another network, the *cwnd* is initialized to one segment. Each time an ACK of a transmitted packet is received, the congestion window is increased by one segment. This provides an exponential *cwnd* growth since the *cwnd* is doubled every RTT (it is not exactly exponential because the receiver may delay its ACKs).

In TCP–Tahoe slow start occurs only if the other end is on a different network.

2.5.2 Packet losses and Fast Retransmit

When data arrives on a large bandwidth link and gets sent out a smaller bandwidth link or when multiple input streams arrive at a router whose output capacity is less than the sum of the inputs, congestion can occur. Assuming that packet loss occurred by data corruption is very small (much less than 1%) the loss of a packet signals congestion somewhere in the network between the source and destination. The packet loss is indicated either by timeout (TO) or the receipt of triple duplicate ACKs (TD). We next describe the TO and TD losses.

TO loss

As mentioned in section 2.2 TCP provides a reliable transport layer. One of the ways it provides reliability is for each end to acknowledge the data it receives from the other end. However, data segments and acknowledgments can get lost. TCP handles this by setting a timeout when it sends data. If the data is not acknowledged before the timeout expires, TCP retransmits the data. A discussion on the implementation of the timeout and retransmission strategy is given in section 2.6 below.

TD loss and Fast Retransmit

The implementation of TCP timeouts may lead to long periods of time during which the connection goes idle while waiting for a timeout to expire. To remedy this, a heuristic mechanism called *fast retransmit* that sometimes triggers the retransmission of a dropped packet sooner than the regular timeout mechanism was added to TCP. The fast retransmit mechanism enhances the timeout facility without replacing the regular timeouts.

The fast retransmit is implemented as follows. Each time a data packet arrives at the receiving side, the receiver responds with an acknowledgment, even if this sequence number has already been acknowledged. Thus, when a packet arrives out of order TCP cannot acknowledge the data the packet contains because earlier data has not yet arrived and TCP resends the same acknowledgment it sent the last time. This second transmission of the same acknowledgment is called a *duplicate ACK*. When the sending side sees a duplicate ACK it knows that the other side must have received a packet out of order, which suggests that an earlier packet might have been lost. Since it is also possible that the earlier packet was delayed rather than lost, the sender waits until it sees some number of duplicate ACKs and then retransmits the missing packet. In practice, TCP waits until it has received three duplicate ACKs before

retransmitting the packet. Such a loss indication is called triple-duplicate ACK loss (TD). In the event that the outstanding data is delayed rather than lost, the receiving side will receive multiple copies of the same packet.

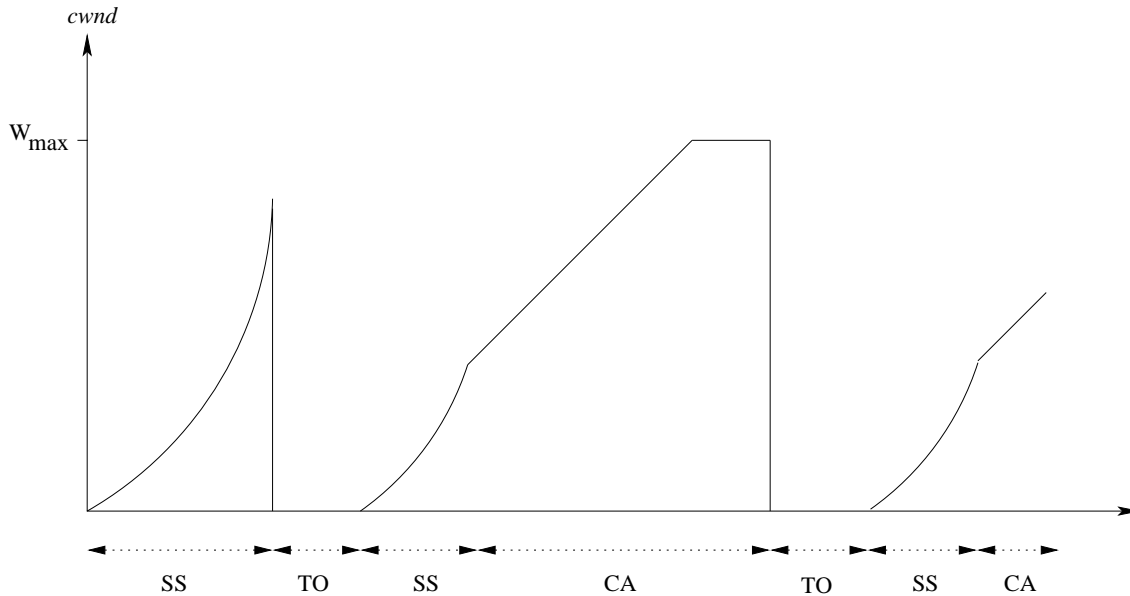
2.5.3 Congestion Avoidance

The congestion avoidance (*CA*) algorithm is used to respond to the congestion that is signalled by packet loss. Both *SS* and *CA* require that the *cwnd* and another variable called the slow start threshold (*ssthresh*) be maintained for each connection. A given connection initializes the *cwnd* to one segment and the *ssthresh* to 65535 bytes which is 64 segments when the segment size is 1024 bytes as is the case in our study. When congestion occurs (indicated by a timeout or the reception of duplicate ACKs), the *cwnd* is reset to one segment and

$$ssthresh = \max(2, \lfloor \min(W_m, cwnd)/2 \rfloor). \quad (2.1)$$

Whenever $cwnd \leq ssthresh$, *SS* is performed and *CA* otherwise. The evolution of *cwnd* in *SS* has been described above. In *CA* *cwnd* is incremented by $1/cwnd$ each time an ACK is received and hence by one segment during each round trip time (regardless how many ACKs are received in that RTT). In *SS* the *cwnd* is increased by the number of ACKs received in a round-trip time (*RTT*). The *CA* *cwnd* increase is additive (linear), compared to the *SS* exponential increase as a function of the *RTT*. The *additive increase* of *cwnd* during *CA* and the *multiplicative decrease* of *ssthresh* is often referred to as the additive increase, multiplicative decrease (AIMD) algorithm (see section 2.1.8 of [61]).

The above TCP-Tahoe algorithms are shown in figure 2.3. As shown in the figure TCP starts in *SS* increasing the *cwnd* size exponentially as a function of *RTT*. If an ACK of a packet is not received within the previously calculated *RTT*, TCP waits for the receipt of triple duplicate ACKs or for the timeout to expire. When a loss is detected by timeout (TO) or triple duplicate ACKs (TD) then the TCP-Tahoe resets *cwnd* to 1, sets the *ssthresh* to maximum of half the previous *cwnd* value and 2 and resumes *SS* until this *ssthresh* value is reached. When the *ssthresh* value is reached TCP starts *CA* where it increases the *cwnd* linearly as a function of *RTT* until the maximum window size is reached after which the *cwnd* is not increased any further. When a loss is detected at this stage, TCP returns to *SS* reducing the *cwnd* to 1 and begins another round of window evolution.

Figure 2.3: TCP-Tahoe *cwnd* evolution

2.6 Round trip time measurement and timeout calculation

The round trip time (*RTT*) is the time from start of the data packet's transmission until the corresponding ACK packet is received. The measurement of the *RTT* experienced on a given connection is fundamental to TCP's timeout and retransmission. The *RTT* can change over time, as routes might change and as network traffic changes. TCP tracks these changes and modifies its timeout value accordingly.

When TCP sends a data segment, it records the time. TCP records the time again when the ACK for that segment arrives. The difference between these two times is a *SampleRTT*. Not all data segments are timed. Most Berkeley-derived implementations of TCP measure only one *RTT* value per connection at a time. If the timer for a given connection is already in use when a data segment is transmitted, that segment is not timed. The timing is done by incrementing a counter every time the 500-ms TCP timer routine is invoked. This means that a segment whose acknowledgment arrives 550-ms after the segment was sent could end up with either a 1 tick *RTT* (implying 500 ms) if it was sent after the previous tick or a 2 tick *RTT* (implying 1000 ms) if it was sent before the previous tick. The timer for a connection is started when the first segment is transmitted, and turned off when its acknowledgment arrives. If for example three TCP clock ticks occur during the period, the *RTT* is measured to be 1500 ms.

After the sender measures a new *SampleRTT*, the timeout (T_0) is calculated as a function of

both the average *RTT* and the mean deviation in that average using the Jacobson/Karels algorithm as follows:

$$\begin{aligned} \textit{Difference} &= \textit{SampleRTT} - \textit{EstimatedRTT} \\ \textit{EstimatedRTT} &= \textit{EstimatedRTT} + (\delta \times \textit{Difference}) \\ \textit{Deviation} &= \textit{Deviation} + \delta(|\textit{Difference}| - \textit{Deviation}) \end{aligned}$$

where δ is a fraction between 0 and 1, *Deviation* is the mean deviation which is a good approximation to the standard deviation, but easier to compute as described by Jacobson. (Calculating the standard deviation requires a square root.)

TCP then computes the timeout value (T_0) as follows:

$$T_0 = \mu \times \textit{EstimatedRTT} + \phi \times \textit{Deviation} \quad (2.2)$$

where based on experience, μ is typically set to 1 and ϕ is set to 4. From the equation it can be seen that when the variance is small, T_0 is close to *EstimatedRTT* and a large variance causes the *Deviation* term to dominate the calculation.

As shown in the source code of the *ns2* simulator [70] and section 2.1.5 of [61], the minimum value of T_0 is $2 \times \textit{tick}$. In many TCP implementations a *tick* equals 500ms yielding a minimum T_0 of 1 second. Other operating systems such as Solaris have smaller tick values (see section 2.1.5 of [61]). In our analytical models where it is impossible to use Equation 2.2 we use

$$T_0 = \max(3 \textit{tick}, 4\overline{RTT}) \quad (2.3)$$

as is the case in [1, 25, 38, 40].

2.6.1 The exponential back-off

In section 2.5.2 we saw that packet losses are detected by TO or TD. If a packet loss is detected by TO the packet is retransmitted by setting its retransmission timeout. The timeout T_i of retransmission i is calculated as

$$T_i = \begin{cases} 2^i T_0, & 1 \leq i \leq 6 \\ 64 T_0, & i \geq 7. \end{cases} \quad (2.4)$$

This doubling of the retransmission timeout (RTO) every time a packet is retransmitted is called *exponential back-off*.

2.6.2 Karn's algorithm

If a transmitted packet times out the timeout value is backed off as shown in Equation 2.4 and the packet is retransmitted with a longer timeout (retransmission timeout) value. If an acknowledgment is received following this retransmission, then it is difficult to know whether the ACK is for the first transmission or the second. This situation is called *the retransmission ambiguity problem*. *Karn's algorithm* specifies that when a timeout and retransmission occur, the RTT estimators are not updated when the acknowledgment for the retransmitted data finally arrives. This is because it is not known to which transmission the ACK corresponds. (Perhaps the first transmission was delayed and not lost, or perhaps the ACK of the first transmission was delayed.) Also, since the data was retransmitted, and the exponential back off has been applied to the RTO, this backed off RTO is reused for the next transmission. A new RTO is not calculated until an acknowledgment for a segment that was not retransmitted is received.

2.7 The TCP sliding window algorithm

The *sliding window* is a mechanism which makes TCP's stream transmission efficient. The sliding window protocols use network bandwidth better as they allow the sender to transmit multiple packets before waiting for an acknowledgement. The sliding window algorithm is described in figure 2.4.

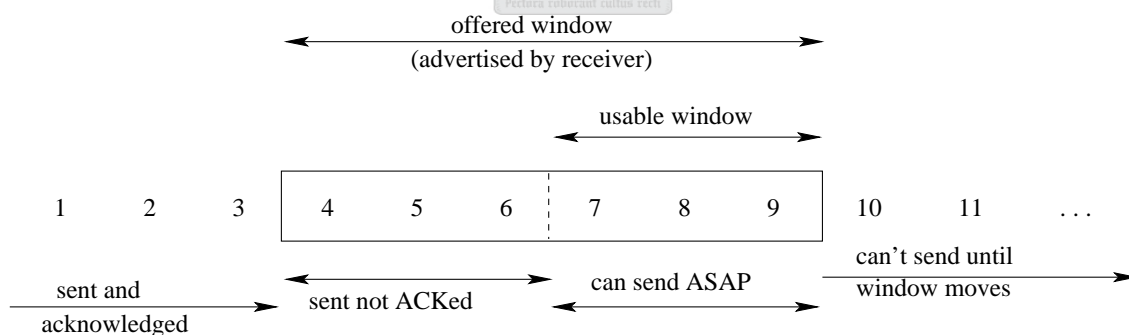


Figure 2.4: Visualization of TCP sliding window

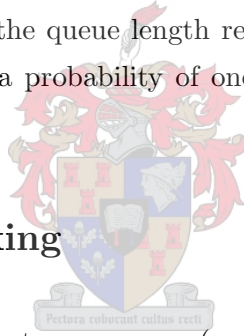
In this figure the bytes are numbered 1 through 11. The window advertised by the receiver is called the *offered window* and covers bytes 4 through 9, meaning that the receiver has acknowledged all bytes up to and including number 3, and has advertised a window size of 6. The sender computes its usable window, which is how much data it can send immediately. Over time this sliding window moves (slides) to the right, as the receiver acknowledges data.

2.8 Queue management in TCP/IP networks

Queue management refers to the algorithms that manage the length of packet queues by dropping packets when necessary or appropriate. From the point of dropping packets, queue management can be classified into *passive queue management* (PQM), which does not employ any preventive packet drop before the router buffer gets full or reaches a specified value and *active queue management* (AQM), which employs preventative packet drop before the router buffer gets full. PQM (e.g. *tail-drop*) is currently widely deployed in the Internet routers. The default AQM scheme is *random early detection* (RED).

The tail-drop scheme drops packets from the tail of a full queue. Packets which are already in the queue are not affected.

In RED a router detects congestion early by computing the average queue length and works with two buffer thresholds Min_{th} and Max_{th} . The router accepts all packets until the queue reaches Min_{th} , a minimum threshold, after which it drops a packet with a linear drop probability distribution function. When the queue length reaches Max_{th} a maximum threshold value, all packets are dropped with a probability of one. For more on queue management techniques see [61].



2.9 Statistical multiplexing

Multiplexing is a concept where a system resource (e.g. a physical link) is shared among multiple users (hosts). If servers send data to clients by sharing a network that contains only one physical link, the flows of data from the servers are multiplexed onto a single physical link by a device (e.g. router) and then demultiplexed into separate flows by another device (e.g. router).

Statistical multiplexing is the most common method for multiplexing flows onto a physical link. Here each flow sends a sequence of packets over the physical link, with a decision made on a packet-by-packet basis as to which flow's packet to send next. If only one flow has data to send, then it can send a sequence of packets back-to-back. However, should more than one flow have data to send, then their packets are interleaved on the link.

The decision as to which packet to send next on a shared link can be made by a router that transmits packets onto the shared link. A router can be designed to service packets on a first-in-first-out (FIFO) basis or the different flows can be serviced in a round-robin manner.

Certain flows can also be made to receive a particular share of the link's bandwidth, or to have their packets never delayed in the router for more than a certain length of time. Such a network which allows flows to request such treatment is said to support *quality of service* (QoS).

2.10 Some basic definitions

2.10.1 Throughput

Definition 2.10.1. *Throughput is the amount of data transferred by a network from a sender to a receiver during a unit of time.*

2.10.2 Bandwidth–delay product

Definition 2.10.2. *The bandwidth–delay product (BDP) defines the amount of data a TCP connection should have “in flight” (data that have been transmitted but not yet acknowledged) at any time to fully utilize the channel available capacity.*

The delay used in this case is the RTT , and the bandwidth is the capacity C of the bottleneck link in the path. Hence the BDP which is the capacity of the link is calculated as

$$\text{capacity (bits)} = \text{bandwidth (bits/sec)} \times \text{round-trip time (sec)}.$$

In cases where the BDP is greater than the maximum allowable TCP window advertisement (65535 bytes), a TCP connection is not able to fill the pipe between the sender and receiver and hence unable to attain the maximum throughput. To solve this problem TCP has a new window scale option to enable it to increase the window size.

2.11 MPLS networks

Choosing the next hop for a packet in an Internet Protocol (IP) is done by partitioning the set of possible packets into a set of forwarding equivalence class (FECs) and choosing the next hop for the FEC.

In conventional IP (Internet Protocol) forwarding, a router considers two packets to be in the same FEC when the network prefixes of their destination addresses are the same. As the packet travels in the network, each router reexamines the packet and assigns it to an FEC.

In MPLS networks the assignment of a packet to an FEC is done only once, when the packet enters the MPLS network. The assignment can be based on a rule that considers not only the destination address field in the packet header but also other fields, as well as information that is not present in the network layer header (for example the port on which the packet arrived). The FEC assigned to the packet is encoded as a short, fixed-length *label*. When a packet is forwarded, the label is sent along with it (packets are labeled before they are forwarded). At subsequent hops, the packet's network layer header is no longer analyzed. Rather, the label carried by the packet is used as an index into a table that specifies the next hop for the packet as well as a new label. For more on MPLS networks see [54].



Chapter 3

A short survey of analytical models of TCP

A number of analytical models have been developed to estimate the performance of TCP connections interacting over a common underlying IP network. This chapter presents a classification of analytical models of TCP and explains the modeling technique we use in the subsequent chapters. A short summary of this chapter is given in section 3.3.

3.1 Classification of analytical models of TCP

Analytical models to estimate the performance of TCP connections can be classified into different groups. The following criteria can be used to classify the TCP models.

3.1.1 The *RTT* and the packet loss probability are known or unknown

Using this criterion, models to estimate the performance of TCP connections can be grouped into two classes:

Models where the *RTT* and the packet loss probability are known

These models assume that the round trip time (RTT), which is the time from the start of data packet's transmission until the time at which the corresponding acknowledgement is received, and the loss characteristics of the IP network are known, and derive from them the throughput and the delay of TCP connections. Models such as [2, 21, 53, 72, 81] belong to this class.

Models where the *RTT* and the packet loss probability are unknown

These models assume that only the basic parameters (network topology, the number of users, data rates, propagation delays, buffer sizes, etc.) are known, and derive from them the performance metrics which directly account for the quality of service. These include the throughput, the queuing delay of TCP connections, the round trip times, the timeout probabilities, the loss characteristics of the IP network and other performance measures.

These models consist of two sub-models (1) a sub-model similar to the models of the first class to account for the TCP congestion control algorithms, and (2) a sub-model that describes the characteristics of the IP network that carries the TCP segments. The two sub-models are jointly solved through an iterative fixed point algorithm (FPA). Models such as [8, 9, 20, 55] belong to this class.

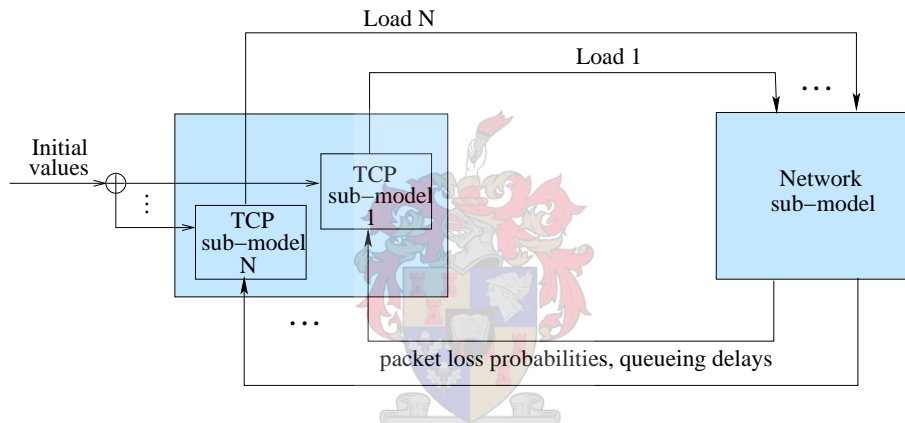


Figure 3.1: The iterative solution procedure with the interaction between the TCP and the network sub-models

Figure 3.1 shows a high-level description of this modeling process. The analytical model has two parts, the TCP sub-model and the network sub-model. The TCP sub-model can be an aggregate of several sub-models, which represent homogeneous groups of TCP connections. Each homogeneous group represents TCP connections sharing common characteristics (the same TCP version, comparable round trip times, similar loss probability, the same value of the maximum window size expressed in packets and so on). Each homogeneous component receives an estimate of the packet loss probability along the TCP connection routes, as well as estimates of the queuing delays at the routers as inputs from the network sub-model. Each component then produces estimates of the load generated by the TCP connections in the group. The network sub-model receives the estimates of the load generated by the different groups of TCP connections as inputs, and computes the loads on the network channels, and the packet loss probabilities and average queuing delays. These are fed back to the TCP

sub-models in an iterative procedure that is stopped when convergence is reached. At the beginning of the FPA the TCP sub-models are initialized.

We next give the second criterion of classification of TCP models.

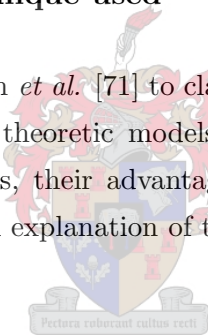
3.1.2 The flows are finite or infinite

TCP connections for the bulk transfer of files and FTP down-loads are called greedy, long-lived, persistent or infinite TCP connections. TCP connections for short file transfer are called finite, non-persistent or short lived. TCP models such as [12, 21, 36, 40, 81] are for finite flows and the majority of models such as [39, 47, 48, 49, 53, 72, 80, 103] are for infinite flows.

The third and last criterion of classification is given below.

3.1.3 The mathematical technique used

This third criterion was used by Olsén *et al.* [71] to classify TCP models into renewal theory models, fixed point models, control theoretic models, processor sharing models and fluid models. A summary of these models, their advantages and disadvantages is given in the following sections. A detailed list and explanation of these models can be found in [71].



Renewal theory models

These models give TCP performance metrics with high accuracy given that the network performance metrics like packet loss probability and delay are known. This approach has lead to many single source models which serve as building blocks for some of the fixed point methods described in the next section. The well-known “PFTK-formula”¹ model ([72]) and the models [21, 47, 49, 53, 58, 80, 103] belong to this class. The advantages and disadvantages of this class of models are given in Table 3.1.

Fixed point models

This category of TCP models represents an advancement from single source models to models of multiple heterogenous TCP sources in arbitrary networks. The fixed point methods

¹Named after the authors last names Padhye, Firoiu, Towsley, Kurose.

Advantages	The throughput is given in a closed form expression which is easy to implement as is the case with the “square root over p law” and the “PFTK-formula” ([72]).
Disadvantages	The models assume that the packet loss rate and the average round trip time are given. The models estimate only the performance for a single sender in the network. Most models are derived under the assumptions of long lived flows.

Table 3.1: Advantages and disadvantages of renewal theory models of TCP

combine the detailed models describing TCP’s behavior with network models resulting in a compound TCP-network model. The packet loss probability and packet delay in the network depend on the sending rate of the arriving traffic, and the flow-controlled TCP sources adjust their sending rates in response to observed packet loss and packet delay. This method of analyzing the interaction between separate models in order to find the network operating regime is called a fixed point method. The models [6, 5, 20, 22, 23, 37, 38, 39, 40, 41, 66, 67, 100, 101] fall into this class. The advantages and disadvantages of the fixed point models are given in Table 3.2.

Advantages	These models combine separate well-examined models for the TCP source and for the network into a compound model. The compound model with basic parameters like network topology and traffic characteristics helps network operators to obtain metrics describing the network and source performance.
Disadvantages	No disadvantages except that the numerical methods used to find the fixed-point sometimes require a great deal of implementation work.

Table 3.2: Advantages and disadvantages of fixed point models of TCP

The other classes of models such as control theoretic approach and the processor sharing approaches could have been put into the class of fixed point methods. However, as shown in [71] the source and network models in the control theory and processor sharing models are not separate interacting models. These source and network models are tightly coupled and they are discussed in separate classes as follows.

Control theoretic models

In control theory, a network with flow-controlled sources is viewed as a large distributed feedback control system. Flow-control is the interaction between the flow-controlled sources and

the network's queues which combine to solve a large optimization problem. An optimization technique of flow control where the sources and the links aim at maximizing total *utilization* under capacity constraints was introduced by Low and Lapsley [58]. The sources update their sending rates as a function of their previous sending rates and the network's congestion signals. They tune their sending rates using their update rule. At the same time the network updates its state and calculates a price (loss rate, delay, etc) as a function of previous prices and the current source rates using the links' update rule. Price information is then signaled back to the sources using a congestion signal. This iterative procedure takes various forms as shown in [58]. It gives an optimal combination of source rates and link prices that maximizes overall welfare, constrained by the network capacity.

Control theoretic approaches are used not only to analyze current source and link protocols but also for designing and proposing new stable and scalable flow-control algorithms for future high capacity networks. In the control theoretic algorithm [7] an optimal link price update interval is crucial for the dynamical network performance. Models such as [44, 51, 57, 59, 60, 74] belong to this class of models. Table 3.3 presents the advantages and disadvantages of this class of models.

Advantages	This approach has proven successful during the design of new flow-control and AQM (Active Queue Management) protocols
Disadvantages	The utility functions derived for TCP-Reno and TCP-Vegas consider only the high level dynamics of the protocols which results in a model that only includes congestion avoidance with no slow start and no timeouts. The analysis does not seem appropriate for modeling the transfer of files from a general file size distribution. Experiments in [71] show that some results of persistent file transfer for TCP-Reno in a Drop-Tail queueing environment are not correct.

Table 3.3: Advantages and disadvantages of control theoretic models of TCP

Processor sharing models

These models give an overview of the expected performance of an underutilized system without having to account for the specific details of the TCP protocol. Given a known arrival intensity and average service requirement for the arrival process these models can be used to derive engineering guidelines for the configuration of the bottleneck link capacity provided the link is not congested. These guidelines can be used to estimate download times for documents of different sizes depending on the core link capacity. Models such as [13, 16, 36, 77, 78, 92]

belong to this class of models. The advantages and disadvantages of this class of models are given in Table 3.4.

Advantages	Suitable for deriving engineering guidelines for the configuration of the bottleneck link capacity using known arrival intensity and average service requirement for the arrival process.
Disadvantages	The packet loss and the requirement for lost packets to be re-sent are not incorporated into the processor sharing TCP–network models. It is not possible to model and distinguish different implementations of TCP protocols and to investigate the performance of the congestion avoidance, slow start, timeout, and exponential back-off mechanisms.

Table 3.4: Advantages and disadvantages of processor sharing models of TCP

Fluid models

These models approximate TCP packets by a fluid flowing through the network. The congestion window size used by the sources and the queue length in the network are assumed to change continuously. The sending rate increases during loss-free periods and decreases in response to loss events. The bulk transfer of files is typically modeled using this technique. A system of stochastic or ordinary differential equations describing the changing rate of the TCP congestion window size and the network queue length are derived. Packet loss in the network is modeled as a stochastic process, and the loss events from this stochastic process control the congestion window size. Hence the evolution of the congestion window becomes coupled to the packet loss process, and TCP performance is derived in terms of the properties of the stochastic loss process. The solutions to the differential equations yield the time evolution of the congestion window size and the network queue length. From the congestion window evolution, performance metrics such as the average throughput are derived. Models such as [2, 3, 15] fall into this class.

The advantages and disadvantages of this class of models are given in Table 3.5.

3.2 Our modeling technique

Based on the first criterion of classification of TCP models presented above, models that do not assume that the RTT and loss probability are known are among the best (most useful) models. The congestion caused by bulk transfer of files and FTP down-loads called greedy,

Advantages	They include the statistical properties of the packet inter-loss process in which case the throughput is derived as a function of the packet loss correlation.
Disadvantages	Even though the correlation among the inter-loss times can be modeled the models are rather simple in that they only model the congestion avoidance phase and the window halving at loss events. Efforts to increase the details of the TCP models by adding maximal windows and timeouts increase the model complexity significantly.

Table 3.5: Advantages and disadvantages of fluid models of TCP

long-lived, persistent or infinite flow TCP connections is considerable. Models of these TCP connections therefore give a good estimation of Internet performance metrics. These TCP models are called models of infinite flows according to the second criterion of classification of TCP models. According to the third criterion of classification, fixed point models are the most useful models. In this thesis we present TCP models of infinite flows which do not assume that the *RTT* and loss probability are known and which can easily be extended to TCP models of finite flows. These models give detailed performance metrics. The mathematical technique we use is the fixed point method.

The modeling technique employed in our study, like that given in [6, 23, 38, 39, 40], involves TCP sub-model(s) and a network sub-model(s). Casetti *et al.* [23] model the behavior of persistent and non-persistent² TCP sources at the application and at the TCP level. The application level is modeled by two states namely *idle* and *active*. In the idle state the application has no active connections and in the active state the application sends data over a TCP connection. Garetto *et al.* [38] adopt the model for window based congestion control protocols by Cigno *et al.* [25] and give a detailed description of the behavior of TCP using an $. / G / \infty$ closed queueing network (see section 4.7) model where each possible TCP state with a specific *cwnd* value is considered as an infinite server queue.

Casetti *et al.* [23] model only the slow start, congestion avoidance and timeout losses of TCP-Tahoe³ at the TCP level. Duplicate ACK losses are not modeled. Only powers of 2 are considered for the values of *ssthresh*. The TCP source model uses average estimates of loss and delay to derive the transition rates and the model assumes that loss rate and delay are exponentially-distributed. Garetto *et al.* [38, 39] avoid these assumptions and model duplicate ACK losses together with the other states of TCP. To account for correlation among losses of

²While persistent connections retain the memory of the protocol status (i.e., the window size, slow start threshold, etc.) when the source is inactive, non-persistent TCP connections reset the protocol status.

³See chapter 2 for the description of these terms.

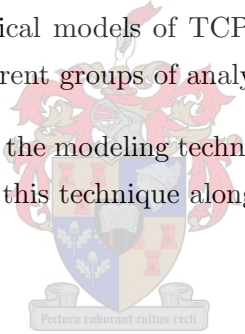
packets within the same congestion window, two different packet loss probabilities, P_{L_f} for loss of the first segment (burst of packets) in the active sliding window and P_{L_a} for any other losses in the same burst, are introduced in [38]. Arvidsson *et al.* [6] describe TCP using an embedded Markov chain model with five transient and one absorbing state.

The TCP models we present in this thesis use the notion of modeling TCP using its flow control algorithms from [6], and the formulas presented in [38, 39, 40]. Our techniques model the steady state operating condition of greedy TCP Tahoe connections. Our models are simple, computationally efficient, accurate and give detailed TCP performance metrics. Our modeling technique can easily be extended to different TCP versions, to short-lived TCP connections in wired and wireless networks.

3.3 Chapter summary

This chapter gives a short survey of analytical models of the TCP/IP protocol, presents different criteria of classifying analytical models of TCP performance, and summarizes the advantages and disadvantages of different groups of analytical models of TCP.

In this chapter we have also explained the modeling technique we have used in the subsequent chapters and motivated our choice of this technique along with the FPA.



Chapter 4

Mathematical background

In this chapter we briefly present some of the mathematical concepts used by the TCP models in the subsequent chapters. We first discuss the fixed point algorithm. We next present the simulation model of this study, some theorems of probability, probability distributions, stochastic processes, queueing theory, queueing networks and the hazard rate function. Finally we give a short summary of this chapter in section 4.9.

4.1 The fixed point algorithm

A common problem in mathematics is solving equations such as $g(x) = c$ where c is a constant. The problem is to find the values of x that satisfy this equation. We can rewrite this equation by subtracting $c - x$ from both sides. This yields $g(x) - c + x = x$. Now define a new function $f(x) = g(x) - c + x$. This is equivalent to

$$f(x) = x. \quad (4.1)$$

An equation of this form is called a *fixed point equation*. To solve it, we have to find values of x that are unchanged i.e. *fixed* by the function f . This can be done by iteration taking an initial guess x_0 for the solution. Now define $x_1 = f(x_0)$, and in turn, for each n , define $x_{n+1} = f(x_n)$.

For example, in a two dimensional plane the fixed points of a function $f(x)$ are the point(s) of intersection of the curve $y = f(x)$ and the line $y = x$.

The above fixed point concept can be extended to a vector-valued function of several variables as shown in Chapter 5. The convergence of the fixed point algorithm (FPA) for analytical models of TCP performance is also discussed in detail in the same next chapter.

4.2 The simulation model

When analyzing the output data from a simulation (see section 11.1 of [11]), a distinction is made between *transient* simulations and *steady-state* simulations. A transient simulation which is otherwise called a terminating simulation is one that runs for some length of time T_E , where E is a specified event (or set of events) which stops the simulation. Such a simulated system begins at time 0 and ends at the stopping time T_E . A steady-state simulation is a simulation whose objective is to study long-run or steady-state behavior of a non-terminating system which is a system that runs continuously, or at least over a very long period of time. For more details and examples of these different types of simulation see chapter 11 of [11].

In this thesis we are interested in the steady-state operating condition of the TCP protocol. The two methods used for eliminating or reducing the deleterious effects of autocorrelation when estimating the mean values in steady-state simulations are the *replication* method and the *batch-means* method. In the replication method a certain number of independent replications are made and in the batch-means method the output data obtained from one replication is divided into batches. As explained in section 11.5.5 of [11] the *batch means* method presented in the next section is a preferable simulation technique for steady-state systems.

4.2.1 The batch means technique of simulation

The batch means technique divides the output data from one replication (after appropriate deletion of the initial transient data) into a few large batches, and then treating the means of these batches as if they were independent. When the raw data after deletion form a continuous-time process, $\{Y(t), T_0 \leq t \leq T_0 + T_E\}$, such as the length of a queue or the level of an inventory, then we form k batches of size $m = T_E/k$ and compute the batch means as

$$\bar{Y}_j = \frac{1}{m} \int_{(j-1)m}^{jm} Y(T_0 + t) dt \quad (4.2)$$

for $j = 1, 2, \dots, k$. In other words, the j th batch mean is the time-weighted average of the process over the time interval $(T_0 + (j-1)m, T_0 + jm)$.

When the raw output data after deletion form a discrete-time process, $\{Y_i, i = d+1, d+2, \dots, n\}$, such as the customer delays in a queue or the cost per period of an inventory system, then we form k batches of size $m = (n-d)/k$ and compute the batch means as

$$\bar{Y}_j = \frac{1}{m} \sum_{i=(j-1)m+1}^{jm} Y_{i+d} \quad (4.3)$$

for $j = 1, 2, \dots, k$ (assuming k divides $n - d$ evenly, otherwise round down to the nearest integer). That is, the batch means are formed as shown below:

$$\underbrace{Y_1, \dots, Y_d}_{\text{deleted}} \underbrace{Y_{d+1}, \dots, Y_{d+m}}_{\bar{Y}_1} \underbrace{Y_{d+m+1}, \dots, Y_{d+2m}}_{\bar{Y}_2} \dots \underbrace{Y_{d+(k-1)m+1}, \dots, Y_{d+km}}_{\bar{Y}_k}$$

Starting with either continuous-time or discrete-time data, the variance of the sample mean is estimated by

$$\frac{S^2}{k} = \frac{1}{k} \sum_{j=1}^k \frac{(\bar{Y}_j - \bar{Y})^2}{k-1} = \frac{\sum_{j=1}^k \bar{Y}_j^2 - k\bar{Y}^2}{k(k-1)} \quad (4.4)$$

where the point estimate \bar{Y} is the overall sample mean of the data after deletion.

If the batch size is sufficiently large, successive batch means will be approximately independent, and the variance estimator will be approximately unbiased. As explained in section 11.5.5 of [11] the recommended batch size $k = 30$.

The 95% confidence interval of the overall mean is given by

$$\left[\bar{Y} - \frac{S}{\sqrt{n}} t_{0.975, n-1}, \bar{Y} + \frac{S}{\sqrt{n}} t_{0.975, n-1} \right] \quad (4.5)$$

where $n = 30$.

4.3 Some basic concepts of probability

In this section we briefly present one definition and two theorems of probability used in our study.

Definition 4.3.1. *For any two events F_1 and F_2 the conditional probability of F_1 given F_2 is*

$$P(F_1|F_2) = \frac{P(F_1 F_2)}{P(F_2)} \quad (4.6)$$

where $P(F_1 F_2)$ denotes the probability that both F_1 and F_2 occur and $P(F_2) > 0$.

Theorem 4.3.1. (The law of total probability): *Let F_1, F_2, \dots, F_n be a partition of a sample space S . Then for any event $E \in S$,*

$$P(E) = \sum_{i=1}^n P(E|F_i)P(F_i). \quad (4.7)$$

The proof can be found in [52].

Theorem 4.3.2. (Bayes' theorem): Suppose F_1, F_2, \dots, F_n are mutually exclusive events such that $\bigcup_{i=1}^n F_i = S$ where exactly one of the events F_1, F_2, \dots, F_n will occur. The probability that F_j occurs given that an event E has occurred is given by the Bayes' formula,

$$P(F_j|E) = \frac{P(E|F_j)P(F_j)}{\sum_{i=1}^n P(E|F_i)P(F_i)} . \quad (4.8)$$

The proof can be found in [79].

4.4 Probability distributions

In this section we present two discrete probability distributions which are frequently used in our study.

4.4.1 Binomial distribution

Suppose that n independent trials, each of which results in a “success” with probability p and a “failure” with probability $1 - p$, are performed. If X represents the number of successes that occur in n trials, then X is said to be a *binomial* random variable with parameters (n, p) and

$$X \sim \text{Bin}(n, p) .$$

The probability mass function of X is given by

$$P(X = x) = \binom{n}{x} p^x (1 - p)^{n-x}, \quad x = 0, 1, \dots, n . \quad (4.9)$$

The expected value of X

$$\mathbf{E}X = np$$

and

$$\text{Var}(X) = np(1 - p) .$$

4.4.2 Geometric distribution

Suppose that independent trials, each having a probability p of being a success, are performed until a success occurs. If we let X be the number of trials required until the first success, then X is said to be a *geometric* random variable with parameter p . Its probability mass function is given by

$$P(X = x) = (1 - p)^{x-1} p, \quad x = 1, \dots, n . \quad (4.10)$$

The expected value of X

$$\mathbf{E}X = \frac{1}{p}$$

and

$$\text{Var}(X) = \frac{1-p}{p^2}.$$

4.5 Stochastic processes

As explained in [79] a *stochastic process* $\{X(t), t \in T\}$ is a collection of random variables. That is, for each $t \in T$, $X(t)$ is a random variable. The index t is often interpreted as time and, as a result, $X(t)$ is referred as the *state* of the process at time t .

The set T is called the *index* set of the process. When T is a countable set the stochastic process is said to be a *discrete-time* process. If T is an interval of the real line, the stochastic process is said to be a *continuous-time* process. For instance, $\{X_n, n = 0, 1, \dots\}$ is a discrete-time stochastic process indexed by the non-negative integers; while $\{X(t), t \geq 0\}$ is a continuous-time stochastic process indexed by the nonnegative real numbers.

The *state space* of a stochastic process is defined as the set of all possible values that the random variables $X(t)$ can assume. Thus, a stochastic process is a family of random variables that describes the evolution through time of some (physical) process.

Definition 4.5.1. A stochastic process $X = \{X_n, n = 0, 1, 2, \dots\}$ is called a *Markov chain* with state space S provided that

$$P(X_{n+1} = x_{n+1} | X_0 = x_0, \dots, X_n = x_n) = P(X_{n+1} = x_{n+1} | X_n = x_n) \quad (4.11)$$

for all $x_0, \dots, x_{n+1} \in S$ and $n = 0, 1, 2, \dots$.

Equation 4.11 may be interpreted as stating that, for a Markov chain, the conditional distribution of any future state X_{n+1} given the past states X_0, X_1, \dots, X_{n-1} and the present state X_n , is independent of the past states and depends only on the present state.

We next briefly present some elementary queueing theory concepts.

4.6 Queueing theory

In this section we present the $M/M/1/K$ and $M/G/\infty$ queueing systems used by the TCP models discussed in subsequent chapters. The queueing system consists of a waiting line of finite or infinite size and one or more identical servers.

4.6.1 The $M/M/1/K$ system

In an $M/M/1/K$ queueing system, the maximum number of jobs in the system is K , which implies a maximum queue length of $K - 1$. An arriving job enters the queue if it finds fewer than K jobs in the system and is lost otherwise. The behavior can be modeled as a birth-death process as shown in section 2.6 of [14].

4.6.2 The $.G/\infty$ system

In the $.G/\infty$ queueing system (see section 22.7 of [102]) the arrival process is not specified and there are an infinite number of servers each having a certain service rate. The service times are described by a general distribution. In this queueing system a customer never has to wait for service to begin.

4.7 Queueing networks

In a queueing network at least two service stations are connected to each other. A station, i.e., a *node*, in the network represents a resource in the real system.

A queueing network is called *open* when jobs can arrive from outside the network at every node and depart from the network from any node. A queueing network is said to be *closed* when jobs can neither enter nor leave the network. The number of jobs in a closed network is constant. A network in which a new job enters whenever a job leaves the system can be considered as a closed network. For more on open and closed queueing networks see [14].

A queueing network can be single class or multi-class. In a single class queueing network there is only one class of customers. A multi class queueing network contains an arbitrary but finite number of different classes of customers. For more on multi class and single class queueing networks see [69].

In [40] a multi class open queueing network is used to model short-lived TCP connections. In [39] and in our case a single class closed network of $.G/\infty$ queues is used.

As in chapter 4 of [52] the analysis of the movement of customers among the network nodes becomes impractical for complex models. However by imposing certain constraints on the scheduling rules and the service time distributions at the network service centers, the network equations can be reduced to a simple form. Queueing networks which conform to these

constraints are called *product form networks*.

Several efficient algorithms for calculating performance measures for closed product-form queueing networks have been developed. The most important ones (see [14]) are the *convolution algorithm* and the *mean value analysis* (MVA). The convolution algorithm is an efficient iterative technique for calculating a normalization constant, which is needed when performance measures are computed. The name of this technique reflects the method of determining the normalization constant, which is similar to the convolution of two probability mass functions (see [96]). In contrast, the mean value analysis is an iterative technique where the mean values of the performance measures can be computed directly without computing the normalization constant. For more on these and other algorithms for calculating the performance measures of the product form networks see [14].

For the single class closed queueing network of $. / G / \infty$ queues of our study we employed MVA to find the TCP performance metrics as shown below.

4.7.1 The mean value algorithm (MVA): A single class closed queueing network of $. / G / \infty$ queues

MVA (see section 8.2 of [14]) is based on *Little's theorem* and the *arrival theorem*. Let \bar{N} , γ , $\bar{\tau}$ and μ denote the mean number of customers, the throughput, the mean response time and the service rate of a node or the overall queueing network system respectively. Little's law states that

$$\bar{N} = \gamma \bar{\tau} \quad (4.12)$$

and the arrival theorem states that

$$\tau = 1/\mu \quad (4.13)$$

for an infinite server (IS) queue which is the case in our study.

The algorithm of the mean value analysis for a network of infinite server (IS) queues then reduces to the simple formulas shown in section 6.1.6. For more on the MVA see section 8.2 of [14].

4.8 The hazard rate function

The hazard function¹ $h(x)$ is the ratio of the probability density or mass function $P(x)$ to the survival function $S(x)$, given by

$$h(x) = \frac{P(x)}{S(x)} = \frac{P(x)}{1 - F(x)} \quad (4.14)$$

where $F(x)$ is the cumulative distribution function corresponding to the probability function $P(x)$.

4.9 Chapter summary

This chapter explains the mathematical concepts used in the following chapters and presents the simulation model we used in our study.



¹also known as the failure rate, hazard rate, or force of mortality (see pp 240 of [79] or [98])

Chapter 5

Convergence of the FPA used to study reliable Internet protocols (TCP)

Analytical models are important tools for the performance investigation of the Internet. As explained in section 3.2 the literature shows that the fixed point algorithm (FPA) (see section 4.1) is the most useful way of solving such analytical models of Internet performance.

Apart from what is observed in experimental literature, no comprehensive proof of the convergence and uniqueness of the FPA is given. In some studies the convergence and uniqueness are shown by assuming some conditions and considering specific scenarios. As also cited in [66] the convergence of FPA for long-lived TCP connections with AQM routers (see section 2.8) was proven in [20]. Garetto *et al.* [66] also gave a proof of convergence for short-lived TCP flows. However these studies consider only a single bottleneck network. In this chapter we show how analytical models of TCP performance converge to a unique fixed point. The basic principles of our proof apply to both single and multiple bottleneck networks, to short and long-lived TCP connections and to both Drop Tail and AQM routers.

This remedies concerns about the convergence of the FPA used in Internet (TCP) modeling and the uniqueness of the fixed point. It also explains the convergence of the FPA observed in the experimental literature and provides a base for further use of the method to develop models of the TCP/IP. The chapter also specifies conditions a modeler should ensure in order to use the FPA for solving reliable Internet protocols like TCP.

The rest of the chapter is organized as follows. We first present a theorem about the convergence of a FPA in section 5.1. In section 5.2 we explain the FPA of analytical models of

TCP performance. In sections 5.3, 5.4 and 5.5 we show the convergence and uniqueness of the usual two dimensional and multi dimensional FPA for single and multi bottleneck links. Summary of the chapter is presented in section 5.6.

5.1 Brouwer's fixed point theorem

One of the oldest fixed-point theorems, *the Brouwer's fixed point theorem* (see [17, 62]) was developed in 1910. Before we state the theorem, we define the terms and concepts used in the theorem.

- If n is a positive integer, then the *ordered n -tuple* used in this chapter is a sequence of n real numbers (a_1, a_2, \dots, a_n) . The set of all ordered n -tuples is called *n -space* and is denoted by \mathbb{R}^n . Let \mathbf{x} denote an element of \mathbb{R}^n .
- An *n -ball* denoted \mathbb{B} (see [94]) is the interior of a sphere in \mathbb{R}^n and is sometimes called the *n -disk*.
- Let a sequence $\{\mathbf{x}_m\}$ of vectors converge to \mathbf{x} in \mathbb{R}^n . The vector \mathbf{x} is the limit vector of the sequence.
- Let $X \subset \mathbb{R}^n$ be the domain of the function \mathbf{f} . The function \mathbf{f} is said to be continuous on the set X if for all $\mathbf{x} \in X$, and every convergent sequence \mathbf{x}_m to \mathbf{x} , $\mathbf{f}(\mathbf{x}_m)$ converges to $\mathbf{f}(\mathbf{x})$.
- A set is *open* if for every point in the set, we can find a small neighborhood such that all points in the neighborhood are within the set. A set is *closed*, if its complement is open.
- A set $A \in \mathbb{R}^n$ is *bounded* if there is some $x \in \mathbb{R}$, the set of real numbers, such that $\|\mathbf{a}\| < x$ for all $\mathbf{a} \in A$.
- A set is *compact* if it is bounded and closed relative to \mathbb{R}^n . Moreover, if \mathbf{f} is a continuous function, then it maps compact sets to compact sets.
- A set S in an n -dimensional space is called a *convex set* (see [95]) if the line segment joining any pair of points of S lies entirely in S .

Theorem 5.1.1. (Brouwer's fixed point theorem) *Every continuous mapping \mathbf{f} of a closed n -ball to itself has a fixed point. Alternatively, let $A \subset \mathbb{R}^n$ be a non empty compact convex set and $\mathbf{f}: A \rightarrow A$ a continuous function. Then \mathbf{f} has a fixed point, i.e. $\mathbf{f}(\mathbf{x}) = \mathbf{x}$ for some $\mathbf{x} \in A$.*

Its proof is omitted and can be found in [30, 62] and the references therein.

5.2 FPA of analytical models of TCP performance

As explained in section 3.1.3 the fundamental concept used when combining the separate TCP and network models in the FPA is as follows. The packet loss probability and packet delay in the network depend on the sending rate of the arriving traffic: the flow-controlled TCP sources adjust their sending rates in response to observed packet loss and packet delay. The TCP sub-models calculate the load offered to the respective network sub-models. The network sub-models in turn calculate the loss probability and queuing delay for the corresponding TCP sub-models in an iterative fixed point procedure (FPA). Hence the usual fixed point elements (entries) used in solving analytical models of TCP performance are the packet loss probability P_L , the average buffer occupancy E_N from which the queuing delay and the RTT are calculated and the load offered by the TCP connections Λ . In addition to these entries one can also have other fixed point entries resulting in a multi dimensional FPA for both single and multi bottleneck links. We next discuss the usual two dimensional FPA for a single bottleneck link.

5.3 The two dimensional FPA for a single bottleneck link

To prove the convergence of the two-dimensional FPA of analytical models of TCP performance for a single bottleneck link using Theorem 5.1.1 (the Brouwer's fixed point theorem) it suffices to show that a *continuous* function \mathbf{f} exists in a *non empty compact convex set* or in a closed n -ball or equivalent such that

$$\mathbf{f}(P_L, E_N) = (P_L, E_N). \quad (5.1)$$

5.3.1 The compact and convex set (region) where the FPA is carried out

The packet loss probability value, P_L is between 0 and 1 and the buffer occupancy value E_N is between 0 and K where $K - 1$ is the router buffer capacity. These two intervals form a non empty compact convex set of points (plane) in which the fixed point procedure is carried out.

We next derive the function \mathbf{f} and show why and how it is continuous for a single bottleneck link.

5.3.2 The continuous fixed point function

To derive the fixed point Equation 5.1 and show that it is continuous, we will first use well known formulas for the derivation of the load that the TCP sub-models offer to the network sub-models. We then prove that these formulas are continuous and conclude that the formula for the throughput (load offered by a TCP connection) is continuous. We will also use the packet loss probability and queuing delay as given by the $M/M/1/K$ queuing system for the network sub-model and show that these functions are continuous. Finally we will construct the vector valued fixed point function given in Equation 5.1 and show that it is continuous.

The formulas for the TCP sub-models and their continuity

TCP connections adjust the load they offer to the network as a function of the packet loss probability and queueing delay at the bottleneck router. This relationship is given by the well known square-root law (see [72]) as follows.

Let P_L denote the packet loss probability at a bottleneck router. Let E_N denote the queue length at the bottleneck router. Let Λ denote the load offered by the TCP sub-model(s). Let d denote the packet queueing delay at the bottleneck router. From Little's Law

$$d = \frac{E_N}{(1 - P_L)\Lambda}. \quad (5.2)$$

Let RTT (see section 6.2.2) denote the average round-trip time. Let b denote the number of packets required for the TCP destination host to send an ACK packet. Usually $b = 1$ or $b = 2$. Let T_0 denote the TCP initial timeout value in seconds.

The throughput of a long-lived TCP connection can be obtained by the *square root formula*,

$$\lambda = s(P_L, RTT) = \frac{1}{RTT} \sqrt{\frac{3}{2bP_L}}. \quad (5.3)$$

However this formula ignores the effects of timeouts (TO). Hence the throughput is given by the more sophisticated *PFTK* formula which takes timeouts under consideration as follows ([73]).

Let W_m denote the maximum TCP window size expressed in packets. Let $W(P_L)$ denote the expected TCP window size as a function of packet loss probability, P_L . Then [73]

$$W(P_L) = \begin{cases} \frac{2}{3} + \sqrt{\frac{4(1-P_L)}{3P_L}} + \frac{4}{9} & W(P_L) < W_m \\ W_m & W(P_L) \geq W_m. \end{cases} \quad (5.4)$$

Let $Q(P_L, w)$ denote the probability that a loss in a window of size w is a *TO*. Then [73]

$$Q(P_L, w) = \begin{cases} 1 & w \leq 3 \\ \frac{(1-(1-P_L)^3)(1+(1-P_L)^3(1-(1-P_L)^{w-3}))}{1-(1-P_L)^w} & P_L \neq 0 \\ \frac{3}{w} & P_L = 0. \end{cases} \quad (5.5)$$

Let $G(P_L)$ denote a polynomial term used in the *PFTK* formula by

$$G(P_L) = 1 + P_L + 2P_L^2 + 3P_L^3 + 4P_L^4 + 5P_L^5 + 6P_L^6. \quad (5.6)$$

Let $E[X]$ denote the expected round number when the first packet loss occurs. A round begins when a packet is sent and ends when its ACK arrives. As shown in [73],

$$E[X] = \begin{cases} W(P_L) & W(P_L) < W_m \\ \frac{W_m}{4} + \frac{1-P_L}{P_L W_m} + 1 & W(P_L) \geq W_m. \end{cases} \quad (5.7)$$

The throughput of a TCP connection is given by the *PFTK* formula

$$\lambda = t(P_L, RTT) = \frac{\frac{1-P_L}{P_L} + \frac{W(P_L)}{2} + Q(P_L, W(P_L))}{RTT(E[X] + 1) + \frac{Q(P_L, W(P_L))G(P_L)T_0}{1-P_L}} \quad (5.8)$$

which takes the timeouts into account. Substituting Equation 5.7 into Equation 5.8 and multiplying the first part of the resulting equation by $(1-P_L)/(1-P_L)$ and the second part by P_L/P_L yields

$$\lambda = t(P_L, RTT) = \begin{cases} \frac{\frac{(1-P_L)^2}{P_L} + \frac{W(P_L)}{2}(1-P_L) + Q(P_L, W(P_L))(1-P_L)}{RTT(W(P_L)+1)(1-P_L) + Q(P_L, W(P_L))G(P_L)T_0}, & W(P_L) < W_m \\ \frac{1-P_L + \frac{W(P_L)}{2}(P_L) + Q(P_L, W(P_L))P_L}{P_L RTT(\frac{W_m}{4} + \frac{1-P_L}{P_L W_m} + 2) + P_L \frac{Q(P_L, W(P_L))G(P_L)T_0}{1-P_L}}, & W(P_L) \geq W_m. \end{cases} \quad (5.9)$$

$W(P_L) < W_m$ implies that

$$\frac{2}{3} + \sqrt{\frac{4(1-P_L)}{3P_L}} + \frac{4}{9} < W_m \quad (5.10)$$

so that

$$P_L > \frac{1}{\frac{3}{4}W_m^2 - W_m + 1}. \quad (5.11)$$

The function $W(P_L)$ given in Equation 5.4 is continuous as

$$\lim_{P_L \rightarrow \frac{1}{\frac{3}{4}W_m^2 - W_m + 1}} W(P_L) = W_m = W\left(\frac{1}{\frac{3}{4}W_m^2 - W_m + 1}\right). \quad (5.12)$$

Similarly it can be shown that $E[X]$ is a continuous function of P_L .

It can be shown using L'Hopital's rule that the function $Q(P_L, W_m)$ described in Equation 5.5 is a continuous function of P_L . The function $Q(P_L, W(P_L))$ is also continuous as $W(P_L)$ is continuous. The polynomial function $G(P_L)$ given by Equation 5.6 is also continuous.

Besides the continuity of $\lambda = t(P_L, RTT)$ is not affected by the value of RTT which is greater than 0.

Therefore the function $\lambda = t(P_L, RTT)$ of persistent TCP Reno connections given by Equation 5.9 is continuous since a combination and composition of continuous functions is continuous at all appropriate points. For other TCP implementations with Drop Tail and RED the throughput formulas given in [49] can be used and the continuity can be similarly shown.

Using similar arguments of the above formulas of persistent TCP connections, the continuity of the square root formula for the rate of non-persistent TCP flows given in [10] can be shown. Any point of discontinuity can be removed by re-defining the function.

We next discuss the continuity of the network sub-model formulas.

The formulas for the network sub-models and their continuity

The network sub-model which focuses on the IP network receives the average traffic load Λ packets/sec collectively offered by the TCP sub-model(s). The network sub-model ($M/M/1/K$) with a router buffer capacity of $K - 1$ packets and a link capacity of C packets/sec (the load $\rho = \Lambda/C$) is used to compute the loss probability P_L and the expected number E_N of customers in the queueing system. The queueing delay part of the RTT is calculated from E_N for the TCP sub-models.

The $M/M/1/K$ queueing system yields the loss probability

$$P_L = \begin{cases} \frac{1-\rho}{1-\rho^{K+1}}\rho^K & \rho \neq 1 \\ \frac{1}{K+1} & \rho = 1 \end{cases} \quad (5.13)$$

and the queue length

$$E_N = \begin{cases} \frac{\rho}{1-\rho} - (K+1)\frac{\rho^{K+1}}{1-\rho^{K+1}} & \rho \neq 1 \\ \frac{K}{2} & \rho = 1. \end{cases} \quad (5.14)$$

A simple way of accounting for the burstiness of TCP traffic using the $M/M/1/K$ is shown in [66] so that the above closed form expressions of P_L and E_N still hold.

Using L'Hopital's rule the quantities P_L and E_N can be shown to be continuous functions $h_1(\Lambda)$ and $m(\Lambda)$. This implies that RTT which is a continuous function $u(E_N)$ is also a continuous function $h_2(\Lambda)$. If there are N TCP connections, the total load offered $\Lambda = N\lambda = Nt(P_L, RTT) = g(P_L, RTT)$ for some continuous function g .

The fixed point formula and its continuity

From the above arguments the fixed point equation used in modeling TCP is given by

$$\begin{aligned}
 (P_L, E_N) &= (h_1(\Lambda), m(\Lambda)) \\
 &= (h_1(g(P_L, RTT), m(g(P_L, RTT))) \\
 &= (h_1(g(P_L, u(E_N)), m(g(P_L, u(E_N))))) \\
 &= (f_1(P_L, E_N), f_2(P_L, E_N)) \\
 &= \mathbf{f}(P_L, E_N).
 \end{aligned} \tag{5.15}$$

Theorem 5.3.1. *The function \mathbf{f} given in Equation 5.15 above is continuous.*

Proof. The functions, h_1, m, u , and g are all shown to be continuous in the preceding sections. Hence the functions f_1 and f_2 which are compositions of continuous functions are also continuous. This implies that the vector valued fixed point function \mathbf{f} given by Equation 5.15 is a continuous function. \square

In section 5.3.1 we have shown that a compact and convex region where the fixed point algorithm for TCP models is carried out exists. In section 5.3.2 we have constructed a fixed point vector valued function for TCP sub-models and shown that it is continuous. We can now use the Brouwer's fixed point theorem to prove the convergence of the fixed point procedure as shown in the following theorem.

Theorem 5.3.2. *The function \mathbf{f} given by Equation 5.15 has a fixed point in the non empty compact convex set explained in section 5.3.1.*

Proof. A direct result of Theorem 5.3.1 and the Brouwer's fixed point theorem. \square

We next show that this fixed point is unique.

5.3.3 Uniqueness of the fixed point of TCP models

To prove the uniqueness of the fixed point of analytical models of TCP, we first construct a fixed point function of the TCP throughput and show that it is continuous and decreasing.

We then state two theorems and prove them. We use these theorems to complete the proof of the uniqueness of the fixed point of the analytical models of TCP.

As shown in [73] and explained in [50] the throughput function given by Equation 5.9 can be expressed as

$$\begin{aligned}\lambda &= t(P_L, RTT) \\ &= \frac{1}{RTT\sqrt{\frac{2P_L}{3}} + 3T_0\sqrt{\frac{3P_L}{8}}P_L(1 + 32P_L^2)}.\end{aligned}$$

This implies that for a single TCP–network sub–model pair with N active TCP connections

$$\begin{aligned}\Lambda = N\lambda &= \frac{N}{h_2(\Lambda)\sqrt{\frac{2h_1(\Lambda)}{3}} + 3T_0\sqrt{\frac{3h_1(\Lambda)}{8}}h_1(\Lambda)(1 + 32(h_1(\Lambda))^2)} \\ &= F(\Lambda)\end{aligned}\tag{5.16}$$

where $RTT = h_2(\Lambda)$ and $P_L = h_1(\Lambda)$ as shown in the previous sections.

If there are k TCP sub–models each of which offers λ_i to the same bottleneck link, let N_i denote the number of active TCP connections in TCP sub–model i . Let D denote the queueing delay and c_i refer to other components of RTT like the propagation delay which are constant for each TCP sub–model. Since D is a continuous function of E_N which in turn is a continuous function of Λ , D is a continuous function $h_3(\Lambda)$. Now we have

$$\begin{aligned}\Lambda &= \sum_{i=1}^k \lambda_i = \sum_{i=1}^k N_i t(P_L, RTT_i) = \sum_{i=1}^k N_i t(P_L, D + c_i) \\ &= \sum_{i=1}^k N_i t(h_1(\Lambda), h_3(\Lambda) + c_i) = H(\Lambda).\end{aligned}\tag{5.17}$$

The first derivative $F'(\Lambda)$ is

$$\begin{aligned}F'(\Lambda) &= -\left(h_2(\Lambda)\sqrt{\frac{2h_1(\Lambda)}{3}} + 3T_0\sqrt{\frac{3h_1(\Lambda)}{8}}h_1(\Lambda)(1 + 32(h_1(\Lambda))^2)\right)^{-2} \times \\ &\quad D_\Lambda\left(h_2(\Lambda)\sqrt{\frac{2h_1(\Lambda)}{3}} + 3T_0\sqrt{\frac{3h_1(\Lambda)}{8}}h_1(\Lambda)(1 + 32(h_1(\Lambda))^2)\right)\end{aligned}$$

where $D_\Lambda(\dots)$ stands for the first derivative of the expression with respect to Λ .

The first derivatives of $h_1(\Lambda) = P_L$ and $h_2(\Lambda) = RTT = u(E_N)$ are positive implying that the functions, h_1 and h_2 are increasing. This can be verified by the fact that when the traffic load increases the loss probability P_L and the queueing delay E_N both increase. Hence $F'(\Lambda) < 0$ for all possible values of Λ . This implies that the function $F(\Lambda)$ is continuous and

decreasing function for $\Lambda > 0$ ($P_L > 0$). This can also be verified by the fact that when the loss probability and queuing delays increase the TCP throughput decreases.

Similarly it can be shown that the fixed point function H used for the many TCP sub-models case is also continuous and decreasing function of Λ .

The following statements which are based on continuous and decreasing functions may be well known facts. However we put them as theorems in order to easily reference them from the succeeding parts of the chapter.

Theorem 5.3.3. *A continuous decreasing function $p_1(x)$ and a continuous increasing function $p_2(x)$ of real numbers intersect at at most one point.*

Proof. Suppose the two functions p_1 and p_2 intersect at two points where x is x_1 and x_2 . This implies that $p_1(x_1) = p_2(x_1)$ and $p_1(x_2) = p_2(x_2)$. Let $x_1 < x_2$. This implies that $p_1(x_1) > p_1(x_2)$ and $p_2(x_1) < p_2(x_2)$ as p_1 and p_2 are decreasing and increasing functions respectively. We then have $p_2(x_1) = p_1(x_1) > p_1(x_2) = p_2(x_2)$. But this is a contradiction as the function p_2 is an increasing function and $x_1 < x_2$. Therefore the two functions meet at at most one point. \square

The following theorem is the result of this theorem.

Theorem 5.3.4. *A continuous decreasing function p of one variable has at most one fixed point.*

Proof. The function $q(x) = x$ is an increasing function. Therefore this function and the decreasing function $p(x)$ intersect at at most one point by Theorem 5.3.3. This in turn implies that the fixed point function $p(x) = x$ has at most one fixed point. \square

The following corollary is the result of Theorem 5.3.4 and Theorem 5.3.2.

Corollary 5.3.1. *Each of the functions F and H given by Equations 5.16 and 5.17 has a unique fixed point.*

Proof. From Theorem 5.3.2 a fixed point exists and from Theorem 5.3.4 there is at most one fixed point. This implies that a unique fixed point exists for the each of the functions F and H . \square

From the above arguments and theorems we have the following important theorem which shows the uniqueness of the fixed point algorithm used to model TCP.

Theorem 5.3.5. *The vector valued function of two variables, \mathbf{f} given by Equation 5.15 has a unique fixed point.*

Proof. Suppose there are two fixed points (P_{L_1}, E_{N_1}) and (P_{L_2}, E_{N_2}) . This implies that there are two fixed points $\Lambda = F(\Lambda)$ and $\Lambda' = F(\Lambda')$ where F is defined in Equation 5.16. But this is a contradiction as the function F has a unique fixed point as shown in Corollary 5.3.1. Hence $(P_{L_1}, E_{N_1}) = (P_{L_2}, E_{N_2})$ and the function \mathbf{f} has a unique fixed point. \square

Theorem 5.3.2 and 5.3.5 show that the usual two dimensional fixed point analytical models of TCP performance converge to a unique point.

5.4 A multi dimensional FPA for a single bottleneck link

In the succeeding chapters of this thesis we use a multi-dimensional fixed point procedure where the variables, the packet loss probability P_L , the average buffer occupancy E_N and the probability that the *ssthresh* value is i $P_t(i)$, $2 \leq i \leq W_m/2$ are the fixed points. W_m is the maximum window size which is taken to be 64 packets in this study.

As explained in the previous sections to prove the convergence of the FPA of the analytical models of TCP performance, we must show that

$$(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2)) = \mathbf{f}(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2))$$

for some continuous function \mathbf{f} in a non empty compact convex set.

The probability values, P_L and $P_t(i)$ are between 0 and 1 and the buffer occupancy value E_N is between 0 and K . These intervals all form the non empty compact convex set in which the fixed point procedure is carried out. We next show why and how the function \mathbf{f} is continuous.

The load offered by the TCP connections Λ (see Equation 6.18) is a continuous function¹ n of P_L , E_N and $P_t(i)$. It is therefore given by

$$\Lambda = n(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2)) \quad (5.18)$$

The loss probability P_L and the buffer occupancy E_N (see Equations 5.13 and 5.14) are continuous functions² say g and h of Λ and are given by

$$P_L = g(\Lambda) = g(n(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2))) \quad (5.19)$$

and

$$E_N = h(\Lambda) = h(n(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2))). \quad (5.20)$$

¹If any discontinuity exists it is removable by redefining the functions (see section 2.4 of [28] for more on removable discontinuity).

²The continuity of these functions can be verified using the L'Hospital's rule explained in chapter 10 of [28].

The $P_t(i)$ values (see Equation B.3) are functions of the loads offered by the loss states which in turn are a continuous function say p of P_L and E_N . Hence

$$P_t(i) = p_i(P_L, E_N) = p_i(g(\Lambda), h(\Lambda)) = m_i(\Lambda) = m_i(n(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2)))$$

for some continuous functions m_i , $i = 2, 3, \dots, W_m/2$.

From the above arguments the $2 + W_m/2 - 1 = 33$ dimensional fixed point equation used in our study is given by

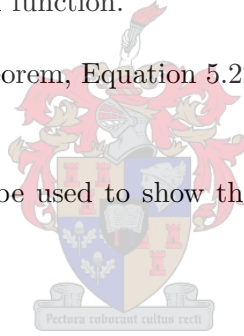
$$(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2)) \tag{5.21}$$

$$\begin{aligned} &= (g(n(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2))), h(n(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2))), \\ &\quad m_2(n(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2))), m_3(n(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2))), \\ &\quad \dots, m_{W_m/2}(n(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2)))) \\ &= \mathbf{f}(P_L, E_N, P_t(2), P_t(3), \dots, P_t(W_m/2)) \end{aligned} \tag{5.22}$$

where \mathbf{f} is a continuous vector valued function.

Now by the Brouwer's fixed point theorem, Equation 5.22 has a fixed point in the non empty compact convex set explained above.

The arguments of section 5.3.3 can be used to show the uniqueness of this 33 dimensional FPA.



5.5 The FPA of analytical models of TCP for a multi bottleneck network

The FPA of analytical models of TCP for the multi bottleneck scenario is more complex. We consider the following two cases to show the convergence and uniqueness of the fixed point algorithm.

Case 1: When only one group of TCP connections represented by one TCP sub-model crosses the bottleneck links (Homogeneous sources)

Let P_{L_ℓ} be the packet loss probability at bottleneck link ℓ . Let D_ℓ which is a continuous function of E_{N_ℓ} , the queue length at bottleneck link ℓ , be the queueing delay at bottleneck link ℓ . Let Λ_1 be the load offered to the first bottleneck link and Λ_ℓ be the load offered to the downstream bottleneck link ℓ in the same path. Let $P_{S_\ell} = 1 - P_{L_\ell}$ be the probability of a successful packet transmission at bottleneck link ℓ . Let \mathcal{P}_ℓ be the multi bottleneck link path

up to and excluding bottleneck link ℓ . The average load offered to bottleneck link ℓ is given by

$$\Lambda_\ell = \prod_{\ell \in \mathcal{P}_\ell} P_{S_\ell} \Lambda_1. \quad (5.23)$$

This equation can be verified as follows. If Λ_2 and Λ_3 are the loads offered to the second and third bottleneck links

$$\begin{aligned} \Lambda_2 &= \Lambda_1 P_{S_1} = \Lambda_1 h(\Lambda_1) = \hbar_1(\Lambda_1) \\ \Lambda_3 &= \Lambda_2 P_{S_2} = \Lambda_1 P_{S_1} P_{S_2} = \hbar_1(\Lambda_1) h(\Lambda_2) = \hbar_1(\Lambda_1) h(\hbar_1(\Lambda_1)) = \hbar_2(\Lambda_1) \end{aligned}$$

where the functions h given by the $M/M/1/K$ is continuous implying that the \hbar_1 and \hbar_2 are continuous functions of Λ_1 . In general we have

$$\Lambda_\ell = \hbar_\ell(\Lambda_1) \quad (5.24)$$

for a continuous function \hbar_ℓ . We also have $P_{L_\ell} = 1 - P_{S_\ell} = \Phi(\Lambda_\ell)$.

As also shown in section 6.3 the average packet loss probability P_L of TCP connections on a path \mathcal{P} is calculated as

$$P_L = 1 - \prod_{\ell \in \mathcal{P}} (1 - P_{L_\ell}) = 1 - \prod_{\ell \in \mathcal{P}} (1 - \Phi(\Lambda_\ell)) = 1 - \prod_{\ell \in \mathcal{P}} (1 - \Phi(\hbar_\ell(\Lambda_1))) = \phi(\Lambda_1) \quad (5.25)$$

where the function Φ given by the $M/M/1/K$ is continuous implying that the function ϕ is also continuous as composition and product of continuous functions is continuous. The corresponding success probability P_S on the path is $1 - P_L$.

The overall delay in the multi bottleneck link path, d is the sum of delays in all bottleneck links in the path and is therefore given by

$$d = \sum_{\ell \in \mathcal{P}} D_\ell. \quad (5.26)$$

Hence

$$\begin{aligned} RTT &= \Gamma(d) = \Gamma\left(\sum_{\ell \in \mathcal{P}} D_\ell\right) = \Gamma\left(\sum_{\ell \in \mathcal{P}} \Omega(E_{N_\ell})\right) = \Gamma\left(\sum_{\ell \in \mathcal{P}} \Omega(\Psi(\Lambda_\ell))\right) \\ &= \Gamma\left(\sum_{\ell \in \mathcal{P}} \Omega(\Psi(\hbar_\ell(\Lambda_1)))\right) = F(\Lambda_1) \end{aligned} \quad (5.27)$$

where the functions, Γ, Ω, Ψ and F are all continuous as the function Ψ can also be defined by the $M/M/1/K$ queueing system.

The convergence and uniqueness of FPA of analytical models of TCP for this case of multi bottleneck links can now be shown as follows.

Let P_{L_i}, E_{N_i} and $\Lambda_i, 1 \leq i \leq n$ be the packet loss probability and the queue length at and the load offered to bottleneck link i .

Now from the above arguments the usual multi dimensional fixed point equation used in modeling TCP is given by

$$(P_{L_1}, P_{L_2}, \dots, P_{L_n}, E_{N_1}, E_{N_2}, \dots, E_{N_n}) = (\Phi(\Lambda_1), \dots, \Phi(\Lambda_n), \Psi(\Lambda_1), \dots, \Psi(\Lambda_n)). \quad (5.28)$$

From Equation 5.24,

$$\begin{aligned} (P_{L_1}, P_{L_2}, \dots, P_{L_n}, E_{N_1}, E_{N_2}, \dots, E_{N_n}) &= (\Phi(\Lambda_1), \dots, \Phi(h_n(\Lambda_1)), \Psi(\Lambda_1), \dots, \Psi(h_n(\Lambda_1))) \\ &= \mathcal{G}(\Lambda_1) \end{aligned} \quad (5.29)$$

where the function \mathcal{G} can be shown to be continuous using the previous arguments and functions. As shown in Equation 5.16 the load offered to the first bottleneck link, Λ_1 is a continuous function of the packet loss probability P_L and the RTT on the path. As shown in Equations 5.25 P_L is a continuous function of the packet loss probabilities P_{L_ℓ} in the path and RTT is a continuous function of the queue lengths E_{N_ℓ} in the path. Thus Equation 5.29 becomes the continuous fixed point function

$$(P_{L_1}, P_{L_2}, \dots, P_{L_n}, E_{N_1}, E_{N_2}, \dots, E_{N_n}) = \mathbf{f}(P_{L_1}, P_{L_2}, \dots, P_{L_n}, E_{N_1}, E_{N_2}, \dots, E_{N_n}). \quad (5.30)$$

The packet loss probability values of each bottleneck link are between 0 and 1 and the buffer occupancy values of each bottleneck link are between 0 and K , where $K - 1$ is the router buffer capacity. These intervals all form a non empty compact convex set in which the fixed point function, \mathbf{f} given by equation 5.30 is defined. By Brouwer's fixed point theorem (Theorem 5.1.1) the multi dimensional fixed point equation 5.30 converges.

The uniqueness of this fixed point is shown using the method of section 5.3.3 and replacing the Λ , P_L and RTT of that section with the Λ_1 , P_L and RTT of this section respectively.

Case 2: When many groups of TCP connections represented by different TCP sub-models cross the bottleneck links (Heterogeneous sources)

In this case we have the same multi-dimensional fixed point equation as in case 1 and is given by

$$\mathbf{f}(P_{L_1}, P_{L_2}, \dots, P_{L_n}, E_{N_1}, E_{N_2}, \dots, E_{N_n}) = (P_{L_1}, P_{L_2}, \dots, P_{L_n}, E_{N_1}, E_{N_2}, \dots, E_{N_n}). \quad (5.31)$$

Using arguments similar to those in case 1 and in section 5.3.2 it can be shown that the function \mathbf{f} given by Equation 5.31 is continuous. Hence a fixed point exists in a non empty compact and convex set which can be constructed by arguments similar to those in case 1 and to those given in section 5.3.1.

We are currently working on the conditions for uniqueness of the FPA of this case. In the following we only show the uniqueness for one specific scenario.

Let a TCP sub-model i cross bottleneck links $1, 2, \dots, k$. Then the total load offered by TCP sub-model i to the first bottleneck link in the path, λ_{i1} is a continuous function ($M/M/1/K$) ψ of $P_{L_1}, P_{L_2}, \dots, P_{L_k}$ and $E_{N_1}, E_{N_2}, \dots, E_{N_k}$. It is given by

$$\begin{aligned}\lambda_{i1} &= \psi(P_{L_1}, P_{L_2}, \dots, P_{L_k}, E_{N_1}, E_{N_2}, \dots, E_{N_k}) \\ &= \varphi(\Lambda_1, \Lambda_2, \dots, \Lambda_k)\end{aligned}\tag{5.32}$$

for some function φ which can be shown to be continuous.

The load Λ_j offered to a bottleneck link j in the path of TCP sub-model i increases if the load offered by TCP sub-model i increases provided that the loads offered by other groups of TCP connections (TCP sub-models) to bottleneck link j are constant. Now for this simple case where Λ_j can be an increasing function, $\vartheta_j(\lambda_{i1}), 1 \leq j \leq k$, Equation 5.32 becomes

$$\begin{aligned}\lambda_{i1} &= \varphi(\vartheta_1(\lambda_{i1}), \vartheta_2(\lambda_{i1}), \dots, \vartheta_k(\lambda_{i1})) \\ &= \Upsilon(\lambda_{i1})\end{aligned}\tag{5.33}$$

for some function Υ .

Now using the technique of section 5.3.3 the uniqueness of the fixed point can be established. That is if there are more than one fixed points, then there will be more than one λ_{i1} satisfying Equation 5.33. But this is a contradiction as the function Υ used in Equation 5.33 can be shown to be a decreasing function (using the throughput formula and the above arguments).

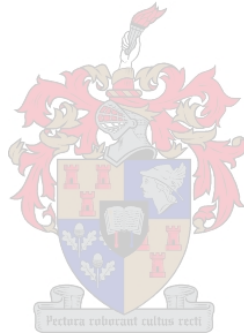
5.6 Chapter summary

In this chapter we have shown how the FPA of analytical models of reliable Internet protocols such as TCP converges to a unique fixed point. The proof of convergence is based on a well known fixed point theorem and the uniqueness proof exploits the feedback and reliable nature of TCP.

We have used well known TCP and network model formulas. We constructed the fixed point functions of TCP performance models of different networking scenarios and proved that each of these functions has a unique fixed point. Unlike the previous works in the literature ([20, 66]), our proof is simple, robust and elegant that its basic principles apply to both short and long lived TCP connections, to single and multi bottleneck links and to AQM and Drop Tail routers.

To show the convergence and uniqueness of the fixed point models of TCP performance we considered the usual fixed points and formulas of the TCP and network models used in the literature. A similar technique can be used to show the convergence and uniqueness of other fixed point models of TCP performance. The theorems used in this chapter specify the conditions one has to ensure in order to use FPA for solving analytical models of reliable Internet protocols like TCP.

Extending the ideas used in this chapter for a deeper analysis of the equilibrium and stability of the TCP protocol and the Internet in general is reserved for future research work.



Chapter 6

The six-state analytical models of TCP

This chapter presents six-state analytical models to estimate the performance of greedy TCP-Tahoe connections. Where no confusion can arise, we use the term TCP to refer to greedy TCP Tahoe connections.

As shown in figure 3.1 the models are composed of TCP and network sub-models which are solved using the FPA discussed in the previous chapters. Each TCP sub-model represents a homogeneous group of TCP connections and each network sub-model represents a bottleneck link traversed by the TCP connections. When all the TCP connections are homogeneous and we have only one bottleneck link, the model reduces to one TCP sub-model and one network sub-model as shown in figure 6.1.

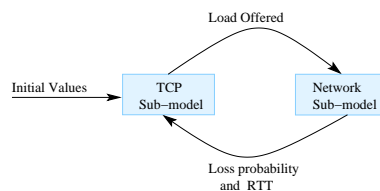


Figure 6.1: The iterative solution procedure

The remainder of this chapter is organized as follows. The TCP and network sub-models are presented in sections 6.1 and 6.2 respectively. The multi bottleneck scenario where many interacting TCP and network sub-models are used is discussed in section 6.3. The analysis of single TCP connections sharing resources with many other TCP connections is explained in section 6.4. The model complexity is described in section 6.5. The chapter summary is given in section 6.6.

6.1 The TCP sub-model

As in [1, 25, 38, 39, 40] we model the dynamics of TCP as described in [83] using a queueing theory framework.

TCP is modeled as a closed network of $.G/\infty$ queues where each queue represents a state of the TCP protocol and each customer represents an active TCP connection. Where no confusion can arise, we use the terms *state* and *queue* interchangeably as each queue in the model represents a state of a TCP connection. A fixed number of active TCP connections (customers) transition within the closed queueing network model. The number of customers at a queue is the number of TCP connections that are in the corresponding state. As there is no limit to the number of connections in a given TCP state, each queue is modeled as an infinite server. The service times are described by a general distribution which depends on the average round trip time experienced by TCP flows and TCP timeouts. The arrival process to the queues need not be specified as only average arrival rates to the states are needed to solve the queueing network and derive the metrics of interest.

Each TCP sub-model can also be treated as a stochastic finite state machine (SFSM), derived from the FSM description of the TCP protocol assigning a probability to the state transitions.

The load generated by the TCP sub-model is derived as follows. We first identify six TCP states. The transition probabilities among the states are calculated using the slow start (*SS*) and congestion avoidance (*CA*) window size distributions and other closed form expressions. Closed form expressions are used to find the probabilities that TCP is in a given state with a specific window size value. These probabilities express the relative number of visits to each queue with specific window sizes. Using these relative visit counts and the retransmission probability distribution, the effects of exponential back-off are obtained. The service times (the times each TCP-connection spends in each state) are next calculated. Using the service times and the relative number of visits, the mean value analysis (MVA) algorithm is used to obtain the normalized packet arrival rate to each queue with a specific window size. These arrival rates together with the average number of packets offered to each queue with a specific window size are used to calculate the load offered to the network sub-model. The *ssthresh* is also calculated from the arrival rates.

As the first step in the calculation of the load offered by a TCP sub-model to the network sub-model we identify the six TCP states in the following section.

6.1.1 The states of TCP connections

Analytical models of greedy TCP flows usually disregard the initial transient phase, assuming that the network quickly operates at a steady-state (see chapter 20.7 of [83]). The states which a TCP connection assumes in a steady-state are

- *SS* which models the TCP slow start state. This refers to both the transmission and retransmission of packets in *SS*.
- *SSF* which models the state where TCP waits for triple duplicate ACKs when losses occur during *SS*.
- *SST* which models the state where TCP waits for a timeout to expire when losses occur during *SS*. This refers to both transmission and retransmission timeouts.
- *CA* which models the congestion avoidance state.
- *CAF* which models losses during *CA* that trigger a fast retransmit.
- *CAT* which models the detection of losses by timeout during *CA*.

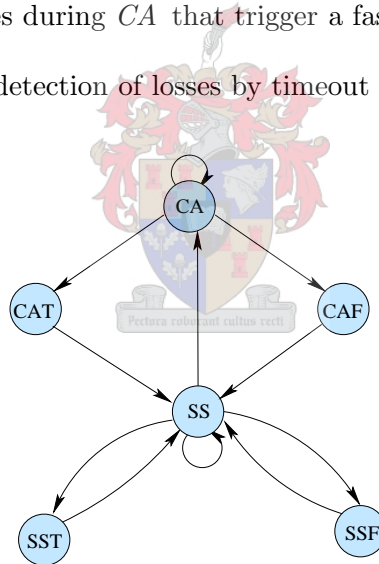


Figure 6.2: The state transition diagram of greedy TCP-Tahoe

Figure 6.2 presents a state transition diagram which is also the queueing network model of TCP-Tahoe as each state is a queue. Each node represents a state of a TCP connection. TCP starts transmission in the *SS* state and remains in that state increasing the *cwnd* value by 1 segment for each successful transmission of a segment thus doubling the *cwnd* value every *RTT* until *ssthresh* is reached. If an ACK signalling success of the segment is not received within one *RTT* then TCP either goes to the *SST* state where it waits for a timeout to expire or to the *SSF* state where it waits for a triple duplicate ACK. When *ssthresh* is reached, TCP

moves into the *CA* state and it remains in that state increasing the *cwnd* by $1/cwnd$ for every ACK received thus increasing the *cwnd* value by 1 every RTT. If TCP is in the *SSF*, *SST*, *CAF* or *CAT* states, it goes back into the *SS* state where the window size is reduced to 1 (TCP-Tahoe) to re-transmit the lost packet. The other transitions can be similarly explained. A detailed description of the transitions is given in appendix B.3.3.

The second step in the calculation of the load offered by a TCP sub-model to the network sub-model computes the transition probability matrix using the *SS* and *CA* window size distributions and other closed form expressions.

6.1.2 The transition probability matrix

This section presents the symbols and equations which are used to derive the transition probability matrix of the above 6 states.

The packet loss probabilities

Let P_L denote the packet loss probability obtained from the network sub-model. As explained in [38, 40, 73] TCP introduces correlations among packet losses that have an impact on performance. The following probabilities are required to account for the correlation among losses of packets transmitted within the same *cwnd*. Let P_{L_f} denote the loss probability of the first packet in the active sliding window¹ and let P_{L_a} denote the loss probability of any other packet in the same window. Let $P_{S_f} = 1 - P_{L_f}$ denote the probability that the first packet in the sliding window is successfully received and let $P_{S_a} = 1 - P_{L_a}$ denote the probability that any other packet in the same window is successfully received.

Consider a group of i packets sent in one *RTT* when $cwnd = w_1 \geq i$ packets. The first packet of the group has a loss probability of P_{L_f} and each of the remaining $i - 1$ packets of that group has a loss probability of P_{L_a} . When an ACK indicating successful transmission of the first packet is received, the window slides one packet forward (see figure B.4). The next packet to be sent from the first group now becomes the first packet of the second group of $cwnd = w_2$ (the active sliding window) with loss probability P_{L_f} . Each of the other packets of w_2 now have a loss probability of P_{L_a} . In *SS* $w_2 = w_1 + 1$ packets. In *CA* $w_2 = w_1$ packets if less than w_1 ACKs are received when $cwnd = w_1$ packets and $w_2 = w_1 + 1$ otherwise².

¹By active sliding window we refer to the current *cwnd* value after the leading edge of the window slides forward.

²*cwnd* is in units of full-size packets

Thus both the *cwnd* and the sliding window protocol (see section 2.7) are represented in our model.

As in [38] we use the relation $P_{L_a} = \alpha P_{L_f}$ where $1 \leq \alpha \leq 1/P_{L_f}$. The factor α is set to \bar{w} (the average congestion window size computed from the TCP sub-model) to model the increased correlation of packet loss with the growth of the window size. The derivation of P_{L_f} is given in appendix A.

The second set of formulas and symbols used in the derivation of the transition probability matrix of the 6 states is given in the next section.

The detailed transition probabilities

Let $\mathcal{T} = \{SS, SSF, SST, CA, CAF, CAT\}$ denote the set of TCP states. Consider a state $A_i \in \mathcal{T}$ and $a_{min} \leq i \leq a_{max}$ where i denotes the size of the congestion window *cwnd*. The values of a_{min} and a_{max} are given in Table 6.1 where W_m is the maximum window size in packets. $W_m = 64$ packets in this study.

	<i>SS</i>	<i>SST</i>	<i>SSF</i>	<i>CA</i>	<i>CAF</i>	<i>CAT</i>
a_{min}	1	1	4	2	4	2
a_{max}	$W_m/2$	$W_m/2$	$W_m/2$	W_m	W_m	W_m

Table 6.1: The maximum and minimum *cwnd* values in each state

Let $P(A_i, B_j)$ denote the probability that TCP goes from state A where *cwnd* = i to state B where *cwnd* = j . Clearly $P(SSF_i, SS_j), P(SST_i, SS_j), P(CAF_i, SS_j)$ and $P(CAT_i, SS_j)$ are all equal to 1 for $j = 1$ and equal to 0 for all other values of j as TCP-Tahoe returns to the SS_1 state every time a packet loss occurs. The remaining $P(A_i, B_j)$ and their derivations are given in appendix B.3.

The third and last set of formulas and symbols used in the derivation of the transition probability matrix concerns the conditional *SS* and *CA* window size distributions. To derive these distributions we used a bounded geometric distribution and a truncated geometric distribution which are discussed in the next section.

The truncated geometric distribution

The probability mass function of the truncated geometric distribution is given by

$$\pi(k) = P(X = k) = \frac{p^{k-1}(1-p)}{1-p^n}, \quad k = 1, 2, \dots, n. \quad (6.1)$$

The geometric distribution (see section 4.4.2 and [97]) is a limiting distribution of the truncated geometric distribution. A generalization of Equation 6.1 where the transition probability p_i from state i into state $i + 1$ is given is presented in the next equation.

The probability mass function of the generalized truncated geometric distribution is given by

$$\pi(k) = P(X = k) = \frac{\prod_{i=1}^{k-1} p_i}{1 + \sum_{j=2}^n \prod_{i=1}^{j-1} p_i}, \quad k = 2, 3, \dots, n \quad (6.2)$$

and

$$\pi(1) = \frac{1}{1 + \sum_{j=2}^n \prod_{i=1}^{j-1} p_i}.$$

The bounded geometric distribution

The probability mass function of the bounded geometric³ distribution is given by

$$\pi(k) = P(X = k) = \begin{cases} p^{k-1}(1-p) & 1 \leq k < n \\ p^{k-1} & k = n. \end{cases} \quad (6.3)$$

A generalization of Equation 6.3 where the probability of success in trial i is p_i is given in the next equation.

The probability mass function of the generalized bounded geometric distribution is given by

$$\pi(k) = P(X = k) = \begin{cases} 1 - p_k & k = 1 \\ \prod_{i=1}^{k-1} p_i(1 - p_k) & 2 \leq k < n \\ \prod_{i=1}^{k-1} p_i & k = n. \end{cases} \quad (6.4)$$

The truncated and bounded geometric distributions are used to derive different models of the *SS* and *CA* algorithms of TCP. The different behaviors of these distributions are investigated in the experiments presented in chapters 7 and 8. It can be seen from the above equations of the distributions that the probability of small values in the bounded geometric distributions

³We call this distribution the bounded geometric distribution to distinguish it from the geometric and truncated geometric distributions.

is smaller (hence the probability of the other values of these distributions larger) than that of the truncated geometric distributions. Our experiments will later show that. Models based on the truncated geometric distribution give a more accurate estimation of small *cwnd* values than models based on the bounded geometric distribution and models based on the bounded geometric distribution give more accurate loss probabilities. Using these models, further analysis of the TCP burstiness (see [46, 90]) can also be made.

The *SS* and *CA* window size distributions

Let $\pi(A)$ and $\pi(A_i)$ denote respectively the probability that TCP is in state A and in state A_i .

In *SS* TCP doubles the *cwnd* every *RTT*, nonetheless *cwnd* grows in a continuous way as shown in appendix (B.2) and figure B.4. Hence before growing to 4, the window must assume the value 3 and keep such value for at least 2 transmission times.

As defined above, the transition probability $P(SS_i, SS_{i+1})$ is the probability of a successful increment of the window size i by 1 in *SS*. The probability of failure to increase the window size i by 1 in *SS* due to packet loss or because *ssthresh* is reached is

$$P_{SS_i} = \begin{cases} 1 & i = W_m/2 \\ 1 - P(SS_i, SS_{i+1}) & 1 \leq i < W_m/2. \end{cases} \quad (6.5)$$

Similarly the transition probability $P(CA_i, CA_{i+1})$ is the probability of successful increment of the window size i by 1 in *CA*. The probability P_{CA_i} of failure to increase the window size i by 1 in *CA* is given by

$$P_{CA_i} = \begin{cases} 1 & i = W_m \\ 1 - P(CA_i, CA_{i+1}) & 2 \leq i < W_m. \end{cases} \quad (6.6)$$

We next present two models of the *SS* window size distribution namely *SS1* and *SS2* and three models of the *CA* window size distribution namely *CA1*, *CA2* and *CA3*. The *SS1* and *CA1* models are based on the generalized bounded geometric distribution. The *SS2* model is based on the generalized truncated geometric distribution. *CA2* is a Markov chain model and has a form of the generalized truncated geometric distribution. The *CA3* model combines the *CA1* and the *CA2* models.

The *SS1* model

The first model of the *SS* window size distribution is based on the generalized bounded

geometric distribution and is given by

$$\begin{aligned}
 P(i \mid SS) &= P(cwnd = i \mid \text{TCP is in state } SS) \\
 &= \prod_{w=1}^{i-1} P(SS_w, SS_{w+1}) P_{SS_i}.
 \end{aligned} \tag{6.7}$$

for $2 \leq i \leq W_m/2$ and $P(1 \mid SS) = P_{SS_1}$ where P_{SS_i} is given in Equation 6.5 above.

The *SS2* model

The second model of the *SS* window size distribution is based on the generalized truncated geometric distribution and is given by

$$\begin{aligned}
 P(i \mid SS) &= \frac{\pi(SS_i)}{\pi(SS)} \\
 &= \frac{\pi(SS_{i-1})}{\pi(SS)} P(SS_{i-1}, SS_i) \\
 &= P(i-1 \mid SS) P(SS_{i-1}, SS_i) \\
 &= \prod_{w=1}^{i-1} P(SS_w, SS_{w+1}) P(1 \mid SS)
 \end{aligned} \tag{6.8}$$

where $2 \leq i \leq W_m/2$ and $\sum_{i=1}^{W_m/2} P(i \mid SS) = 1$.

Thus

$$P(1 \mid SS) + \sum_{i=2}^{W_m/2} \prod_{j=2}^i P(SS_{j-1}, SS_j) P(1 \mid SS) = 1$$

so that

$$P(1 \mid SS) = \frac{1}{1 + \sum_{i=2}^{W_m/2} \prod_{j=2}^i P(SS_{j-1}, SS_j)}.$$

The three models of the *CA* window size distribution are given below.

The *CA1* model

The first model of the *CA* window size distribution *CA1* is based on the generalized bounded geometric distribution and on the fact that for long-lived TCP connections at equilibrium, the *ssthresh* is reached quickly that TCP spends most of the time on the *CA* states. Based on this assumption the few transitions from *SS* into *CA* are not considered in the derivation of the *CA* window size distribution. However these transitions from *SS* into *CA* are accounted for in the derivation of the stationary probability that TCP is in a given state as shown in the next section. Numerical results given in chapter 7 show that the above assumption has a small effect only in the *cwnd* size and *ssthresh* distributions for a small number of active TCP

connections sharing the same link. The $CA1$ model is given by

$$\begin{aligned} P(i | CA) &= P(cwnd = i | \text{TCP is in state } CA) \\ &= \prod_{w=2}^{i-1} P(CA_w, CA_{w+1}) P_{CA_i} \end{aligned} \quad (6.9)$$

for $3 \leq i \leq W_m$ and $P(2 | CA) = P_{CA_2}$ where P_{CA_i} is given in Equation 6.6 above.

The $CA2$ model

The second model of the CA window size distribution is a Markov chain model having a form of the generalized truncated geometric distribution and is given by

$$\begin{aligned} P(i | CA) &= \frac{\pi(CA_i)}{\pi(CA)} \\ &= \frac{\pi(CA_{i-1})P(CA_{i-1}, CA_i) + \pi(SS_i)P(SS_i, CA_i)}{\pi(CA)} \\ &= P(i-1 | CA)P(CA_{i-1}, CA_i) + \frac{\pi(SS_i)}{\pi(CA)}P(SS_i, CA_i). \end{aligned}$$

From figure 6.2

$$\begin{aligned} \pi(CA) &= \pi(SS)P(SS, CA) + \pi(CA)P(CA, CA) \\ &= \pi(SS) \left(P(SS, CA) + \frac{\pi(CA)}{\pi(SS)}P(CA, CA) \right) \\ &= \pi(SS) \left(P(SS, CA) + \frac{\pi(CA)}{\pi(SS)} \left(\sum_{j=2}^{W_m-1} \frac{\pi(CA_j)P(CA_j, CA_{j+1})}{\pi(CA)} + \frac{\pi(CA_{W_m})P(CA_{W_m}, CA_{W_m})}{\pi(CA)} \right) \right) \\ &= \pi(SS) \left(P(SS, CA) + \sum_{j=2}^{W_m-1} \frac{\pi(CA_j)}{\pi(SS)}P(CA_j, CA_{j+1}) + \frac{\pi(CA_{W_m})}{\pi(SS)}P(CA_{W_m}, CA_{W_m}) \right). \end{aligned}$$

Therefore for $i > 2$

$$\begin{aligned} P(i | CA) &= P(i-1 | CA)P(CA_{i-1}, CA_i) \\ &\quad + \frac{P(i | SS)P(SS_i, CA_i)}{P(SS, CA) + \sum_{j=2}^{W_m-1} \frac{\pi(CA_j)}{\pi(SS)}P(CA_j, CA_{j+1}) + \frac{\pi(CA_{W_m})}{\pi(SS)}P(CA_{W_m}, CA_{W_m})} \end{aligned} \quad (6.10)$$

and

$$P(2 | CA) = \frac{P(2 | SS)P(SS_2, CA_2)}{P(SS, CA) + \sum_{j=2}^{W_m-1} \frac{\pi(CA_j)}{\pi(SS)}P(CA_j, CA_{j+1}) + \frac{\pi(CA_{W_m})}{\pi(SS)}P(CA_{W_m}, CA_{W_m})}$$

where for $j > 2$

$$\begin{aligned} \frac{\pi(CA_j)}{\pi(SS)} &= \frac{\pi(CA_{j-1})P(CA_{j-1}, CA_j) + \pi(SS_j)P(SS_j, CA_j)}{\pi(SS)} \\ &= \frac{\pi(CA_{j-1})}{\pi(SS)}P(CA_{j-1}, CA_j) + P(j | SS)P(SS_j, CA_j) \end{aligned}$$

and

$$\frac{\pi(CA_2)}{\pi(SS)} = P(2 | SS)P(SS_2, CA_2).$$

In the above derivation $P(SS, CA)$ is calculated using Equation 6.14 of section 6.1.3 and the same equation is also used to derive $P(CA, CA)$. $P(i | SS) = 0 = P(SS_i, CA_i)$ for $W_m/2 < i \leq W_m$ as the maximum TCP SS window value at equilibrium is $W_m/2$.

The CA3 model

Combining CA1 and CA2 we get the third model of the CA window size distribution

$$\begin{aligned} P(i | CA) &= \frac{\pi(CA_i)}{\pi(CA)} \\ &= \frac{\pi(CA_{i-1})P(CA_{i-1}, CA_i) + \pi(SS_i)P(SS_i, CA_i)}{\pi(CA)} \\ &= P(i-1 | CA)P(CA_{i-1}, CA_i) + \frac{\pi(SS_i)}{\pi(CA)}P(SS_i, CA_i) \\ &= P(i-1 | CA)P(CA_{i-1}, CA_i) + P(i | SS)\frac{\pi(SS)}{\pi(CA)}P(SS_i, CA_i). \end{aligned} \quad (6.11)$$

All the terms in Equation 6.11 are known except $\pi(SS)/\pi(CA)$. Now

$$\frac{\pi(SS)}{\pi(CA)} = \frac{1 - P(CA, CA)}{P(SS, CA)} \quad (6.12)$$

where

$$P(SS, CA) = \sum_{i=2}^{W_m/2} P(i | SS) P(SS_i, CA_i). \quad (6.13)$$

The recursive Equation 6.11 is computed as follows

STEP 0. Use Equation 6.9 to compute initial values for the window size distribution $P(i | CA)$.

STEP 1. Compute

$$P(CA, CA) = \sum_{i=2}^{W_m-1} P(i | CA) P(CA_i, CA_{i+1}) + P(W_m | CA) P(CA_{W_m}, CA_{W_m}).$$

STEP 2. Use Equations 6.11, 6.12 and 6.13 and step 1 to compute the $P(i | CA)$.

Repeat **STEP 1** and **STEP 2** until the values of the $P(i | CA)$ converge.

Using different combinations of the above window size distributions, we have four different models of TCP namely **Stb1**, **Stb2**, **Stb3** and **Stb4** as shown in Table 6.2. There is no noticeable difference in the numerical results obtained by using CA2 and CA3. We therefore use CA3 for **Stb2** and **Stb4**.

	CA1	CA2 / CA3
SS1	Stb1	Stb2
SS2	Stb3	Stb4

Table 6.2: The four analytical models

Now that the formulas and symbols necessary for the derivation of the transition probabilities for the six states are defined above, we next give the transition probability matrix.

The probability $P(A, B)$ that a TCP connection which was in state (queue) A at time n , moves into state (queue) B at time $n + 1$ is

$$\begin{aligned}
P(A, B) &= P(X_{n+1} = B \mid X_n = A) \\
&= \sum_{j=b_{min}}^{b_{max}} P(X_{n+1} = B_j \mid X_n = A) \\
&= \sum_{j=b_{min}}^{b_{max}} \frac{P(X_{n+1} = B_j, X_n = A)}{P(X_n = A)} \\
&= \sum_{i=a_{min}}^{a_{max}} \sum_{j=b_{min}}^{b_{max}} \frac{P(X_{n+1} = B_j, X_n = A_i)}{P(X_n = A)} \\
&= \sum_{i=a_{min}}^{a_{max}} \sum_{j=b_{min}}^{b_{max}} \frac{P(X_n = A_i)}{P(X_n = A)} P(X_{n+1} = B_j \mid X_n = A_i) \\
&= \sum_{i=a_{min}}^{a_{max}} \sum_{j=b_{min}}^{b_{max}} \frac{\pi_{A_i}}{\pi(A)} P(A_i, B_j) \\
&= \sum_{i=a_{min}}^{a_{max}} \sum_{j=b_{min}}^{b_{max}} p(i \mid A) P(A_i, B_j)
\end{aligned} \tag{6.14}$$

where the values of $a_{min}, a_{max}, b_{min}$ and b_{max} for each queue are given in Table 6.1, the $p(i \mid SS)$ and $p(i \mid CA)$ are calculated from the SS , and CA models presented above, and the $P(A_i, B_j)$ are given in appendix B.3.

The transition probability matrix is given in Table 6.3 where it can be seen that $P(SST, SS) = P(SSF, SS) = P(CAT, SS) = P(CAF, SS) = 1$ as TCP-Tahoe always goes back to SS when a timeout (TO) or triple duplicate (TD) loss occurs. The other transition probabilities in the matrix are calculated using Equation 6.14.

As the third step in the calculation of the load offered by a TCP sub-model to the network sub-model we next present the equilibrium probabilities.

	<i>SS</i>	<i>SST</i>	<i>SSF</i>	<i>CA</i>	<i>CAF</i>	<i>CAT</i>
<i>SS</i>	$P(SS, SS)$	$P(SS, SST)$	$P(SS, SSF)$	$P(SS, CA)$	0	0
<i>SST</i>	1	0	0	0	0	0
<i>SSF</i>	1	0	0	0	0	0
<i>CA</i>	0	0	0	$P(CA, CA)$	$P(CA, CAF)$	$P(CA, CAT)$
<i>CAF</i>	1	0	0	0	0	0
<i>CAT</i>	1	0	0	0	0	0

Table 6.3: The transition probability matrix

6.1.3 The equilibrium probabilities

The stationary probabilities $\pi(B)$ that a TCP connection is in state $B \in \mathcal{T}$, which is the set of TCP states described in section 6.1.1, are computed by solving the system of 6 linear equations

$$\pi(B) = \sum_A \pi(A) P(A, B)$$

and normalizing the solution. The system of linear equations can be solved analytically. Thus

$$\pi(CA) = \pi(SS)P(SS, CA) + \pi(CA)P(CA, CA) = \frac{\pi(SS)P(SS, CA)}{1 - P(CA, CA)}.$$

Since TCP has to be in one of the states at a given time

$$\pi(SS) + \pi(SST) + \pi(SSF) + \pi(CA) + \pi(CAT) + \pi(CAF) = 1.$$

Thus

$$\begin{aligned} \pi(SS)(1 + P(SS, SST) + P(SS, SSF)) + \frac{\pi(SS)P(SS, CA)}{1 - P(CA, CA)}(1 + P(CA, CAT) + P(CA, CAF)) &= 1 \\ \pi(SS) \left(1 + P(SS, SST) + P(SS, SSF) + \frac{P(SS, CA)}{1 - P(CA, CA)}(1 + (1 - P(CA, CA))) \right) &= 1 \\ \pi(SS) \left(1 + P(SS, SST) + P(SS, SSF) + P(SS, CA) + \frac{P(SS, CA)}{1 - P(CA, CA)} \right) &= 1 \\ \pi(SS) \left(1 + (1 - P(SS, SS)) + \frac{P(SS, CA)}{1 - P(CA, CA)} \right) &= 1. \end{aligned}$$

Therefore

$$\pi(SS) = \frac{1 - P(CA, CA)}{P(SS, CA) + (2 - P(SS, SS))(1 - P(CA, CA))}.$$

In the above calculations $P(CA, CA) \neq 1$ and hence the denominators are not 0 as it is assumed that long-lived TCP connections congest the bottleneck links that there is at least a loss of one packet when TCP is in the *CA*. Otherwise the equations can be redefined

to remove any discontinuity so that the FPA is used to solve the models. The other four stationary probabilities are given by

$$\begin{aligned}\pi(SST) &= \pi(SS)P(SS, SST) \\ \pi(SSF) &= \pi(SS)P(SS, SSF) \\ \pi(CAT) &= \pi(CA)P(CA, CAT) \\ \pi(CAF) &= \pi(CA)P(CA, CAF).\end{aligned}$$

The $\pi(B_i)$ can now be obtained as follows.

$$\begin{aligned}\pi(SS_i) &= \pi(SS)P(i | SS), \quad 1 \leq i \leq W_m/2 \\ \pi(CA_i) &= \pi(CA)P(i | CA), \quad 2 \leq i \leq W_m\end{aligned}$$

As can be seen from figures B.3 and B.4 of appendix B, TCP offers one packet in SS_1 and 2 packets in each of the SS_i states where $2 \leq i \leq W_m/2$ states. As detailed in appendix B.3.3, the loss of the first packet sent in SS_i is detected either by timeout (TO) or by triple duplicate ACKs (TD) when the window size has grown to $2i - 2$ provided *ssthresh* is not reached until the window grows to $2i - 2$. The loss of the second packet sent in SS when the window size is i is detected when the window size is $2i - 1$ provided *ssthresh* is not reached until the window grows to $2i - 1$.

For example consider *ssthresh* ≥ 4 . If packet 1 is sent and ACK 1 returns, *cwnd* = 2 and two packets 2 and 3 are sent back-to-back. If ACK 2 doesn't return, the loss of packet 2 is detected when *cwnd* = $2 \times 2 - 2 = 2$. If ACK 2 returns, *cwnd* = 3 and packets 4 and 5 are sent. If at this stage ACK 3 doesn't return, the loss of packet 3 is detected when *cwnd* = $2 \times 2 - 1 = 3$. Thus the loss which is detected when the window size is i provided *ssthresh* is not reached until the window grows to i is the loss of the first packet which was sent when the window size was $(i + 2)/2$ if i is even or the loss of the second packet which was sent when the window size was $(i + 1)/2$ if i is odd.

On the other hand even though the losses of the first and second packets sent when the window size was $(i+2)/2+k$, $0 \leq k \leq (i-2)/2$, i being even and $(i+1)/2+k$, $0 \leq k \leq (i-1)/2$, i being odd respectively are detected when the window size is $2((i+1)/2+k)-1 = 2((i+2)/2+k)-2 = i + 2k$, transitions from $SS_{(i+2)/2+k}$ and $SS_{(i+1)/2+k}$ into SST_i or SSF_i occur if there is a loss and *ssthresh* is reached when *cwnd* = i .

Therefore

$$\pi(SST_i) = \sum_{j=a}^i \pi(SS_j)P(SS_j, SST_i)$$

for $1 \leq i \leq W_m/2$ and

$$\pi(SSF_i) = \sum_{j=a}^i \pi(SS_j) P(SS_j, SSF_i)$$

for $4 \leq i \leq W_m/2$ where

$$a = \begin{cases} (i+1)/2, & i \text{ is odd} \\ (i+2)/2, & i \text{ is even.} \end{cases}$$

A transition into CAT_2 occurs only from CA_2 . Hence

$$\pi(CAT_2) = \pi(CA_2) P(CA_2, CAT_2).$$

For all other values of i a transition into CAT_i or CAF_i occurs if either the first packet sent in CA_i is lost or the first packet sent in CA_{i-1} is successful and any of the other packets sent in CA_{i-1} is the first lost packet. For details on these transitions see appendix B.3.3.

Therefore for $3 \leq i \leq W_m$

$$\pi(CAT_i) = \pi(CA_{i-1}) P(CA_{i-1}, CAT_i) + \pi(CA_i) P(CA_i, CAT_i)$$

and for $4 \leq i \leq W_m$

$$\pi(CAF_i) = \pi(CA_{i-1}) P(CA_{i-1}, CAF_i) + \pi(CA_i) P(CA_i, CAF_i).$$

Using the above equilibrium probabilities (which can also be treated as relative visit counts to each queue) and a retransmission probability distribution given below we present the effects of exponential back-off in the next section as the fourth step in the calculation of the load offered by a TCP sub-model to the network sub-model.

6.1.4 The effects of exponential back-off

Unlike the model of Garetto *et al.* [39], in our model the exponential back-off and the other retransmission states are built into the SS and SST states as shown in figures 6.2 and B.1 to reduce the complexity and to obtain simple closed form expressions for the SS and CA *cwnd* size distributions and other values.

After a timeout loss in both SS and CA , TCP returns to SS where the timeout value is obtained using Karn's algorithm (see section 2.6.2) during each retransmission. Define the first transmission to be retransmission 0. A TCP connection which starts in SS with $cwnd = 1$ goes to the $(j+1)$ st retransmission where $0 \leq j < c$ if the j th retransmission is not successful and stays in the j th retransmission if the j th retransmission is successful, where c is the maximum number of consecutive retransmissions allowed before closing the connection. $c = 16$ for TCP-Tahoe. The probability that TCP is in the j th retransmission, $0 \leq j \leq c$

is modeled as a bounded geometric distribution (see section 6.1.2) with parameter P_{S_f} . Let P_{rt_j} denote the probability that TCP is in the j th retransmission. Then

$$P_{rt_j} = \begin{cases} (1 - P_{S_f})^j P_{S_f} & 0 \leq j < c \\ (1 - P_{S_f})^j & j = c. \end{cases} \quad (6.15)$$

In our future work we will see if a truncated geometric distribution (see section 6.1.2) gives a better result.

Let $\pi(SST_i^j)$ be the probability that TCP is in the SST state when the window size is i , $1 \leq i \leq 3$ in retransmission j , $0 \leq j \leq c$. Then

$$\pi(SST_i^j) = P_{rt_j} \pi(SST_i) \quad (6.16)$$

The effect of Karn's algorithm and exponential back-off ends and the retransmission index is reset to zero if the two newly transmitted packets in SS_2 after the successful retransmission are also successful. If transmission of the first or only the second packet is not successful, TCP goes to the SST_2 or SST_3 state of that retransmission where the timeout value is obtained from the exponential back-off using Karn's algorithm. After this timeout value expires TCP goes back to SS_1 of the same retransmission. Thus only the $cwnd$ values 1, 2 and 3 are required to deal with TCP's exponential back-off. This is shown in figure B.2 where the transmission and retransmission states are separately drawn.

The service times of each queue are given in the next section as the fifth step in the calculation of the loads offered by a TCP sub-model to the network sub-model.

6.1.5 The service times

Let $\tau(A_i)$ denote the time which TCP spends in state A_i . These service times and their derivations are given in appendix C with the exception of $\tau(SST_i)$ which is

$$\tau(SST_i) = \begin{cases} \sum_{j=0}^c \pi(SST_i^j) \tau(SST_i^j) & 1 \leq i \leq 3 \\ \tau(SST_i^0) & i > 3. \end{cases}$$

where $\tau(SST_i^j)$ is service time of the SST state when $cwnd = i$ in retransmission j . These values are also given in appendix C.

For $1 \leq i \leq 3$, $\tau(SST_i)$ is the weighted average of the service times of the SST_i 's at each retransmission where the weights are the probabilities of being in retransmission j ($0 \leq j \leq c$)

with the corresponding window size. The same idea is used in the derivation of the load offered by SST_i given in section 6.1.7 below.

Using the service times and the relative visit counts explained above, the mean value analysis algorithm (MVA) is used as shown in the next section to obtain the normalized arrival rate to each queue with a specific *cwnd* size. This is the sixth step in the calculation of the load offered by a TCP sub-model to the network sub-model.

6.1.6 Mean value analysis

As discussed in the previous sections we model TCP using a closed queueing network of infinite server queues with N customers (flows, active TCP connections). The mean value analysis (see Section 4.7.1) of this queueing network has the simple form

$$\begin{aligned}\gamma &= N / \sum_{A \in \mathcal{T}} \sum_{i=a_{min}}^{a_{max}} \pi(A_i) \tau(A_i) \\ \gamma(A_i) &= \pi(A_i) \gamma \\ N(A_i) &= \gamma(A_i) \tau(A_i)\end{aligned}$$

where $N(A_i)$, $\tau(A_i)$ and $\gamma(A_i)$ are the average number of connections, the time spent by a TCP connection and the throughput (see Definition 2.10.1) at queue A_i respectively and γ is the common throughput. Since the queueing network is closed

$$\lambda(A_i) = \gamma(A_i) \tag{6.17}$$

where $\lambda(A_i)$ is the arrival rate of connections at queue A_i .

At this stage the *ssthresh* is calculated as shown in section B.3.2 of the appendix. Finally the arrival rates of connections to each queue calculated above and the number of packets offered by each TCP queue with a specific *cwnd* size are used to calculate the load offered by a TCP sub-model to the network sub-model as shown in the next section.

6.1.7 The load offered to the network sub-model

The aggregate packet load offered to the IP network by the TCP connections in the different states is

$$\Lambda = \sum_{A \in \mathcal{T}} \Lambda(A) = \sum_{A \in \mathcal{T}} \sum_{i=a_{min}}^{a_{max}} \lambda(A_i) L(A_i) \tag{6.18}$$

where $\lambda(A_i)$ is given in Equation 6.17 and $L(A_i)$ is the number of packets offered from queue A_i to the underlying IP network. The $L(A_i)$ are used in [39] and their derivations are given

in appendix D with the exception of $L(SST_i)$ which is given by

$$L(SST_i) = \begin{cases} L(SST_i^0) & i = 1, 2 \quad \text{and} \quad i > 3 \\ \sum_{j=0}^c \pi(SST_i^j) L(SST_i^j) & i = 3 \end{cases}$$

where $L(SST_i^j)$ is the load offered by the SST state when the $cwnd = i$ in retransmission j . These values are also given in appendix D. We next explain the network sub-model which receives the load Λ from the TCP sub-model(s).

6.2 The network sub-model

The TCP sub-model which is discussed in the preceding sections requires the loss probabilities P_{L_f} and P_{L_a} and \overline{RTT} of TCP connections as inputs. If these parameters can be found from network measurements or other sources, the TCP sub-model can predict the performance measures such as the offered load, throughput, time spent in timeout, $cwnd$ and $ssthresh$ distributions. If these parameters are not given then our models can predict the loss probabilities, the \overline{RTT} , the average buffer occupancy and the other performance metrics mentioned above given only a physical description of the network using basic metrics such as the network topology, link capacity and buffer size. This is done using a fixed point procedure containing the TCP and network sub-models as explained in the previous sections.

The network sub-model which focuses on the IP network receives the traffic load Λ packets/sec collectively offered by the TCP connections in different states (queues) from the TCP sub-model. The network sub-model with a router buffer capacity of $K - 1$ packets and a link capacity of C packets/sec (the load $\rho = \Lambda/C$) is used to compute the packet loss probability P_L and the expected number E_N of customers in the queueing system from which the queueing delay part of the RTT is calculated for the TCP sub-models. The simple $M/M/1/K$ queue is used to compute the loss probability P_L and the \overline{RTT} for long-lived TCP connections as the loss rate of greedy connections is dominated by the TCP protocol behavior that defines the traffic load. In this study only drop-tail buffers are considered. As explained in [39] the analysis of IP networks using Active Queue Management (AQM) algorithms (see section 2.8) such as RED [34] does not pose additional problems. This is because when RED is used the estimation of the average buffer occupancy allows the estimation of the packet drop probability, and the average buffer occupancy is easy to compute. Besides the smaller loss correlation imposed by AQM schemes simplifies the estimation of the network behavior. The $M/M/1/K$ yields the loss probability, P_L given by Equation 5.13.

6.2.1 The average buffer occupancy

We used two different techniques to obtain the expected number E_N of customers in the router buffer namely the $M/M/1/K$ technique which uses Equation 5.14 and the *geometric technique*. We include the geometric technique in this study because it shows a similar trend of buffer occupancy as simulation as shown in section 7.8. Moreover the geometric technique is interesting as it stems from a simple physical description of the router buffer using the exponential and linear window growth in *SS* and *CA* respectively. Besides when dealing with large buffer sizes and small or large values of ρ the geometric technique is faster than the $M/M/1/K$ technique.

The geometric technique

As shown in [84] and simulation experiments such as figure 3 of [38], and figure 9 of [4] the buffer occupancy of TCP has triangular emptyings that separate short intervals in which the buffer is full. This is shown in figure 6.3. In the figure K is the router buffer capacity (number of packets waiting and in service) and $N(t)$ is the number of packets in the router buffer at time t . From the figure, it can be seen that the buffer for a fixed number of active TCP connections fills up through time as each connection increases its window size. When the buffer is full, packets are dropped. But the ACKs sent by the receiver before the buffer was full cause the sources to send more packets to refill the buffer. Such packet transmission of the same window size continues for one \overline{RTT} after which the first loss is detected and the sources reduce their window sizes. Hence the buffer remains full for $a = \overline{RTT}$. After this time the sources react to the loss and the buffer decreases to its minimum value until it gets new packets. After the TCP sources detect the losses they reduce their window sizes to 1 (which is the case with TCP-Tahoe). Then the buffer fills up again at least when the connections are greedy. We assume that the average window size has already grown to 2 when the buffer is at its minimum value as the TCP sources start sending packets when they realize that the queue length is decreasing.

Assume that the buffer becomes full when the average *cwnd* of each TCP connections is \overline{w} packets. This implies that the total number of packets sent by a TCP connection from the time the buffer is at its minimum ($cwnd = 2$) until the time the buffer fills up ($cwnd = \overline{w}$) is $\overline{w} - 2$. This means that $\frac{\overline{w}-2}{2}$ ACKs which all cause each source to send $\frac{\overline{w}-2}{2}$ packets are received by each connection before the buffer becomes full. This raises the total number of packets sent by a TCP connection to $2 + \frac{\overline{w}-2}{2} + \frac{\overline{w}-2}{2} = \overline{w}$. The $\frac{\overline{w}-2}{2}$ unacknowledged

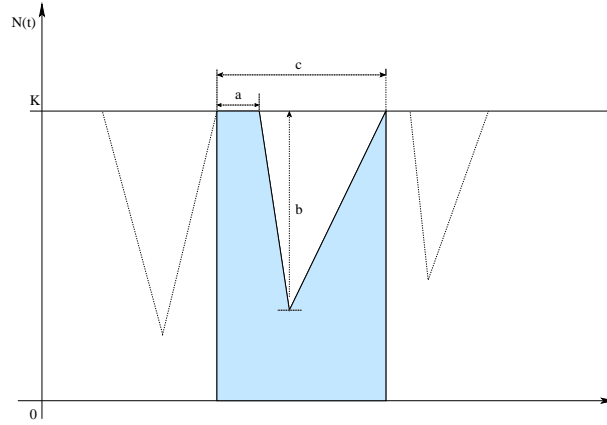


Figure 6.3: The buffer occupation of TCP for a fixed number of concurrent TCP connections

packets sent by each TCP connection fill the buffer. Hence the N active TCP connections send $\frac{N(\bar{w}-2)}{2} = b$ packets to fill the buffer up.

To get the value of c consider figure 6.4 which is similar to figure 21.8 of [83] and figure 2 of [48]. Let W be the *cwnd* size of the previous cycle⁴. In other words a loss occurred when *cwnd* was W segments in which case the *ssthresh* was then set to $W/2$ segments and the *cwnd* was set to 1 segment. One segment is then sent at time 0 and its ACK is returned at time at \overline{RTT} . In the figure it can be seen that for $A \leq \text{cwnd} \leq \frac{W}{2}$, $f(t) = 2^t$ as TCP doubles its window size every RTT in *SS*. Let t_1 be the time TCP spends in *SS* and let t_2 be the time TCP spends in *CA* before the buffer is full. Then we have the following.

$$W/2 = 2^{t_1}$$

so that $t_1 = \log_2(W/2) = \log_2(W) - 1$.

As shown in figure 6.4 we assume that $W/2 \leq \bar{w} \leq W$. This means that the *cwnd* exceeds the *ssthresh* at the time the network is congested (buffer is full). Therefore taking the average, $\bar{w} = (W + W/2)/2 = 3W/4$. Now for $W/2 \leq \text{cwnd} \leq \bar{w}$, $f(t) = t$ as TCP increases linearly every RTT in *CA*.

This implies that the second time interval $t_2 = 3W/4 - W/2 = W/4$.

Therefore the interval between the times when the buffer level starts to decrease and the time the buffer becomes full again ($\text{cwnd} = \bar{w}$) is

$$c - a = (t_1 + t_2)RTT = (\log_2(W/2) + W/4) RTT.$$

⁴Here as used in [47] *cycle* refers to the interval between consecutive reductions of the sending window following a TO or TD loss detection.

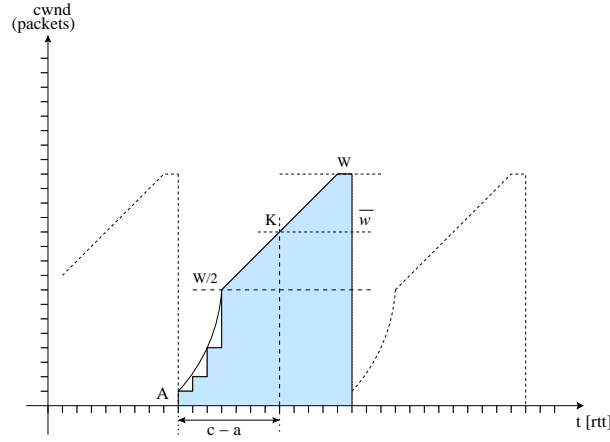


Figure 6.4: The window evolution for a TCP connection

From the previous explanation $a = RTT$. So we have

$$c = (2 + \log_2(\bar{w}/3) + \bar{w}/3) RTT.$$

From figure 6.3 the average buffer occupancy E_N which is the expected number of customers in the router buffer, can now be approximated by taking a fraction of the router buffer capacity K using the shaded area, A_{shaded} and the total area of the rectangle, A_{tot} as follows.

$$\begin{aligned}
 E_N &= \frac{A_{shaded}}{A_{tot}} \times K \\
 &= \frac{Kc - \frac{1}{2}(c-a)b}{Kc} \times K \\
 &= K - \frac{b}{2} + \frac{ab}{2c}.
 \end{aligned} \tag{6.19}$$

Now that the buffer occupancy can be calculated using Equation 6.19, we can derive a simple expression for the packet loss probability using the $M/M/1/K$ queue as follows.

From Equation 5.14 for $\rho \neq 1$ we have

$$E_N = \frac{\rho}{1-\rho} - (K+1) \frac{\rho^{K+1}}{1-\rho^{K+1}}$$

so that

$$\frac{\rho^K}{1-\rho^{K+1}} = \left(\frac{\rho}{1-\rho} - E_N \right) \frac{1}{(K+1)\rho}. \tag{6.20}$$

From Equation 5.13 of the $M/M/1/K$ and using Equation 6.20 for $\rho \neq 1$ we have

$$\begin{aligned}
 P_L &= \frac{1 - \rho}{1 - \rho^{K+1}} \rho^K \\
 &= (1 - \rho) \left(\frac{\rho}{1 - \rho} - E_N \right) \frac{1}{(K + 1)\rho} \\
 &= \frac{(E_N + 1)\rho - E_N}{(K + 1)\rho}
 \end{aligned} \tag{6.21}$$

where E_N is obtained using Equation 6.19.

When dealing with large or small ρ and large buffer sizes K Equations 6.19 and 6.21 may be preferable to Equations 5.13 and 5.14 which have power factors.

6.2.2 The average round trip time

The average round trip time \overline{RTT} is the sum of four different types of delays which are described below.

1. The *time d spent in the queuing system*. From Little's Law it is given by Equation 5.2.
2. The *propagation delay δ* . This is the time it takes one bit to propagate from the sender to the receiver. It is given by $\delta = D/S$ where D is distance of the client nodes from the servers and S is the propagation speed which is equal to the speed of light. This speed depends on the physical medium of the link. It is $3 \times 10^5 \text{ km/sec}$ in a vacuum, $2.3 \times 10^5 \text{ km/sec}$ in a cable, and $2 \times 10^5 \text{ km/sec}$ in a fiber. In this study only fiber links are analyzed.

Our model considers N concurrent TCP connections (flows) from different sources. As given in section 7.2, we assume that the total distance of the TCP transmitters from the receivers is uniformly distributed between D_{min} and D_{max} . Hence the total propagation delay of the TCP flows is also uniformly distributed between δ_{min} and δ_{max} . Since the bandwidth shared by each TCP connection is inversely proportional to the RTT of the connection as shown in [36], a harmonic mean of the propagation delay δ_i of each TCP connection is used to calculate the “average” propagation delay δ used in our model and in [39]. This is given by

$$\frac{1}{\delta} = \frac{1}{N} \sum_{i=1}^N \frac{1}{\delta_i}. \tag{6.22}$$

The propagation delay δ is multiplied by 2 to account for the delay of the ACKs on the reverse path.

3. The *transmission delay* (store-and-forward delay) d_t is the time for all bits of a packet to be transmitted. It is given by $d_t = L/C$ where L is the length (size) of packet in bits and C is the bandwidth (link capacity) in *bits/sec*. In the forward direction a user packet of size 1024 bytes and in the return path an ACK packet of size 40 bytes are considered.
4. The *processing delay* d_p . This is the time needed by a router to check for errors, to determine the destination address, and to calculate which interface (link) to use. It is constant for a given router, depending on a specific router's processing capacity and assumes that the router has sufficient capacity.

6.3 The multi bottleneck scenario

In the preceding sections we explained how a single TCP sub-model interacts with a network sub-model. In the analysis of a real network where TCP connections cross many bottleneck links, an open queueing network of $(M/M/1/K)$ servers is used to model the links. The load of each link is given by the sum of the loads generated by groups of TCP connections sharing the link. Each group of TCP connection constitutes a TCP sub-model. The traffic offered by link ℓ to downstream links is computed by thinning the traffic arriving at the buffer of link ℓ by a factor $1 - P_{L_\ell}$, in order to account for losses where P_{L_ℓ} is the average packet loss probability at the buffer of link ℓ . The $M/M/1/K$ queueing networks are solved separately starting from the ones closer to the sources, and proceeding along the paths followed by the various groups of TCP flows. If the network is not strictly feed-forward where the nodes can be both transmitters and receivers, the traffic arriving at some queues has to be approximated using results obtained in the previous step of the fixed point algorithm (FPA). The convergence of the FPA guarantees that this method is correct.

The average packet loss probability of TCP connections on a path \mathcal{P}_i is calculated using the loss probabilities in the bottleneck links of \mathcal{P}_i whose level of congestion depends on the total number of TCP connections competing for resources on the links. At the network level, we assume that losses in different buffers are not correlated, which is reasonable as there is no coordination between routers in terms of packet losses. Therefore, as given in [41] for each TCP group in path i the probability that a packet is successfully transferred through the network is

$$P_{S_i} = \prod_{\ell \in \mathcal{P}_i} (1 - P_{L_\ell}). \quad (6.23)$$

The corresponding packet loss probability on path i is $1 - P_{S_i}$.

6.4 Individual TCP connections

In the preceding sections we have seen how performance metrics of many interacting TCP connections can be obtained. Here we show how results can be obtained for a single TCP connection competing with other TCP connections.

Thanks to the negligible numerical complexity of our models, we treat the single TCP connection as one TCP sub-model and the group of other TCP connections sharing the same link as another TCP sub-model. We then solve the fixed point procedure. Using this method, results for many individual TCP connections can be obtained in one run of the fixed point procedure by considering each single TCP connection as a TCP sub-model. Figure 3.1 can show this. If the coefficient of variation of connection lengths is small, the TCP connections can be put into one TCP sub-model otherwise different TCP sub-models can be used as shown in the figure.

Our modeling technique solves the limitation of the method used in [39] in finding results for single TCP connections. In [39] the iterative fixed point solution is based on the use of average propagation delay only. Our technique uses a single TCP sub-model for connections having similar propagation delays.

6.5 The complexity of the models

The solution of our models is obtained by a fixed point algorithm using the TCP and network sub-models. In each iteration of the fixed point procedure, only closed form expressions are used to solve the TCP and network sub-models. The complexity of each iteration is dominated by the solution of the TCP sub-models which collectively offer a traffic load to the network sub-model. To find end-to-end TCP performance metrics the TCP and network sub-models are solved for each bottleneck link. Let L denote the number of bottleneck links and let M denote the number of distinct TCP sub-models for each bottleneck link. The complexity of our model for the multi bottleneck network is $O(LM)$ as the simple closed form formulas have to be used for each TCP sub-model at each bottleneck link. The CPU time required by our models for each step of the fixed point procedure, for single bottleneck link and for one TCP-network sub-model pair is small (milliseconds on a 1.6GHz AMD Athlon XP 2000+ PC running Linux).

The number of iterations of the fixed point algorithm before convergence depends on the accuracy ϵ and on the initial settings of the parameters. We used $\epsilon = 10^{-6}$. For the first

point of the solution of the model for a given setup and for a given number of concurrent connections, initially $P_L = 0.01$ and $P(ssthresh = i) = P_t(i) = 2/W_m$ as given in [38].

The CPU time of the *ns2* simulation increases significantly with the increase in the number of active TCP connections, whereas in our model it is almost independent of the number of active TCP connections. For thousands of TCP connections *ns2* simulation experiments cannot be computed on a PC.

In the Polito model (we refer to the model in [39] as the Polito model), a system of N_Q linear equations has to be solved at each iteration of the fixed point procedure for each TCP-sub model at each bottleneck link where N_Q is the number of queues (states) used in the model. N_Q depends on W_m the value of the maximum window size and c the number of consecutive retransmissions allowed before a TCP connection closes. The complexity of Polito model for a multi-bottleneck (realistic) network model is therefore $O(LMN_Q^3)$. The value of M for a realistic network model is large as different connections have different values of RTT , maximum window size, maximum segment size, packet loss probabilities, routes, versions of TCP and other characteristics which are used to classify them into groups of TCP sub-models. Our model can treat each TCP connection as a separate TCP sub-model and hence overcomes the limitation (due to computational complexity) of the method used in [39] to find results for single TCP connections for simple network models.

In real network topologies each TCP sub-model has to be solved for each bottleneck link. Detailed results as a function of many different numbers of active TCP connections are required. In the Polito model, large systems of linear equations have to be solved at every iteration of the fixed point algorithm, for each number of active TCP connections, for each TCP tick value, for each buffer size, for each TCP sub model and for each link in order to get a good picture of the network performance. Therefore the gain in CPU speed of our model is significant.

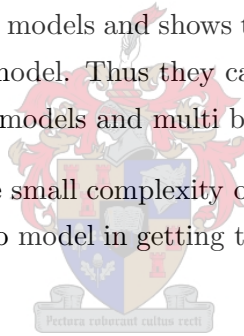
6.6 Chapter summary

This chapter

- presents 6-state analytical models of TCP performance namely, *Stb1*, *Stb2*, *Stb3* and *Stb4* based on a closed network of $. / G / \infty$ queues,
- discusses two network models namely the $M/M/1/K$ technique and the *geometric* technique which stems from a simple physical description of the router buffer using the ex-

ponential and linear *cwnd* growth in *SS* and *CA*, shows the merits of these two models and gives a method of combining them,

- shows how the multi dimensional FPA discussed in the previous chapters can be used to combine the TCP sub-models and an open network of network sub-models,
- shows how simple to solve the MVA becomes for a closed network of $.G/\infty$ queues,
- shows how the packet loss correlation and the sliding window algorithm of the TCP protocol can be modeled,
- shows how the geometric, generalized bounded and truncated geometric distributions can be used to model TCP/IP networks,
- gives closed form expressions for many important TCP performance values and distributions such as the conditional *CA* and *SS* distributions and the packet retransmission probability distribution,
- quantifies the complexity of our models and shows that our models are computationally more efficient than the Polito model. Thus they can be easily applied to a networking scenarios with many TCP sub-models and multi bottleneck links and
- presents a way of exploiting the small complexity of our modeling approach in order to solve the limitation of the Polito model in getting the throughput and other metrics for individual TCP connections.



Chapter 7

Numerical Results: Single bottleneck link analysis

This chapter presents numerical results when the TCP models presented in chapter 6 are applied to evaluate the performance of a single bottleneck link in a European network topology.

The chapter is organized as follows. The validation methodology is presented in section 7.1. The topology of the European network model is presented in section 7.2. The loss probability, the average *cwnd* and *ssthresh* sizes, the *cwnd* and *ssthresh* distributions, the probability that TCP waits for timeout to expire, the buffer occupancy and throughput of individual connections obtained using the *Stb1*, *Stb2*, *Stb3* and *Stb4* models with the $M/M/1/K$ and geometric buffer formulas are presented in sections 7.3 through 7.9. We compare these models with the Polito model [39] and against results from *ns2* simulation experiments. Summary of the chapter is given in section 7.10.

7.1 The validation of the models

The performance predictions obtained from the analytical models of TCP presented in this thesis were compared against point estimates obtained from *ns2* [70] simulation experiments. The “batch means” technique of simulation using 30 batches (see section 4.2.1) was used. The network topology used for the validation of a single bottleneck link case of this chapter and multi bottleneck link case of the next chapter is given in the following section.

7.2 The topology of the Internet-A European network

We validated our model using a network topology which was presented in [23, 38, 39, 40]. It closely resembles the path followed by Internet connections from the Politecnico di Torino (Italy) LAN to sites in Europe and the USA. The network topology is shown in Figure 7.1. On the left, a set of clients is connected to the internal LAN of the Politecnico di Torino. The

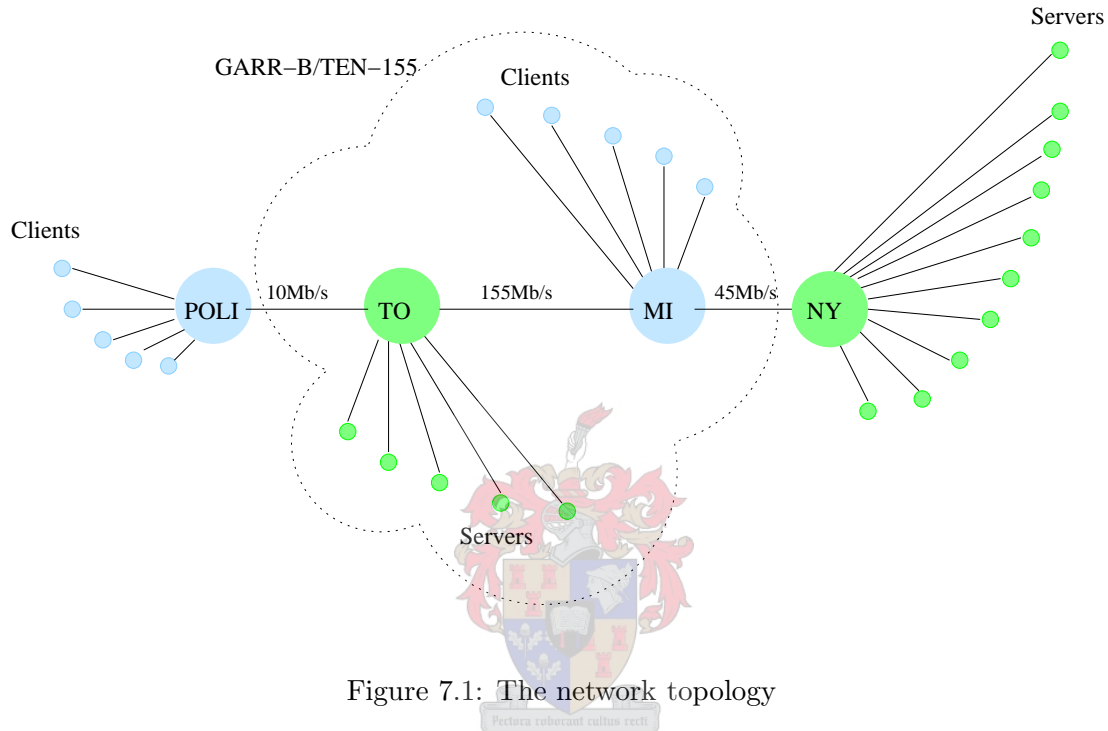


Figure 7.1: The network topology

distance of these clients from the Politecnico router is assumed to be uniformly distributed between 1 and 10 km. The LAN is connected to the Italian IP network for universities and research institutions, named GARR-B, which is part of the European TEN-155 academic network, through a 10 Mb/s link whose length is some 50km. The GARR-B/TEN-155 network consists of a number of routers and 155 Mb/s links. One of these links connects the router in Turin with a router in Milan. This link length is some 100 km. Through the GARR-B/TEN-155 network, clients at the Politecnico can access a number of European servers, whose distance from Politecnico is assumed to be uniformly distributed between 100 and 6,800 km. From Milan, a 45 Mb/s link whose length is some 5,000 km reaches New York and connects the GARR-B network to the North American Internet backbone. Many other clients use the router in Milan to reach servers in the USA. The distance of the clients from Milan is assumed to be uniformly distributed between 200 and 2,800 km. The distance of the servers in the USA from the router in NY is assumed to be uniformly distributed between 200 and 3,800 km. For simplicity, one-way TCP is assumed with an uncongested backward path,

so that ACKs are not lost or delayed. Even though all distances are assumed to be uniformly distributed, it must be noted that our models are not limited to these distributions. Three types of TCP connections are considered:

1. TCP connections from USA servers to Politecnico di Torino clients. The total lengths of these connections vary from 5,351 to 8,960 km.
2. TCP connections from GARR-B/TEN155 servers to Politecnico di Torino clients. The total lengths of these connections vary from, 151 and 6,860 km.
3. TCP connections from USA servers to clients connected to the GARR-B/TEN155 network through the MI-NY link. The total lengths of the connections vary from 5,400 and 11,600 km.

Given the link capacities, congestion occurs in either the 10 Mb/s POLI-TO link, which is crossed by TCP connections of types 1 and 2 or the 45 Mb/s MI-NY link which is crossed by connections of types 1 and 3, or both. The user packet size is assumed to be 1024 bytes and the ACK packet size is assumed to be 40 bytes. The maximum window size is assumed to be 64 user packets. Two buffer sizes of 128 and 512 packets and two TCP tick¹ values of 100ms and 500ms are considered. Connections of type 1 that cross both links are assumed to be 25% of all connections traversing the POLI-TO link. When the POLI-TO link is congested and the MI-NY link is lightly loaded, the scenario is called LOCAL ACCESS, since the bottleneck is only in the local access link. When the MI-NY link is congested and the POLI-TO link is lightly loaded, the scenario is called USA BROWSING, since the bottleneck is only in the transoceanic link. In both cases, the topology reduces to a single bottleneck network. When both links are congested, the topology is a two-bottleneck network. In this chapter we consider the POLI-TO single bottleneck link for the throughput estimation and the MI-NY single bottleneck topology for estimation of all other performance metrics. The next chapter presents an analysis of the two-bottleneck network.

7.3 The loss probability

Figures 7.2 and 7.4 present the packet loss probabilities obtained using the *Stb1*, *Stb2*, *Stb3* and *Stb4* models as a function of the number of active TCP connections when the $M/M/1/K$ buffer formula is used on the MI-NY link. These figures show the loss probabilities for the standard TCP tick value of 500ms and a tick value of 100ms respectively.

¹See section 2.6.

Figures 7.5 and 7.6 show the loss probabilities when the geometric buffer formula is used for TCP tick values of 500ms and 100ms respectively. The buffer capacities used in these experiments are 128 and 512 packets.

Comparison with *ns2* simulations shows that the *Stb1* and *Stb2* models usually give a good estimation of the loss probability when the standard TCP tick value of 500ms and a TCP tick value of 100ms are used with the $M/M/1/K$ and geometric buffer formulas. The *Stb3* and *Stb4* models and the Polito model also give an accurate estimation of loss probability when a TCP tick value of 100ms is used with the $M/M/1/K$ and geometric buffer formulas. When the standard TCP tick value of 500ms is used the *Stb3*, *Stb4* and the Polito models underestimate the loss probability. Table 7.1 identifies the best models for computing the loss probabilities. This comparison is based on how closely the respective models replicate *ns2* experiments.

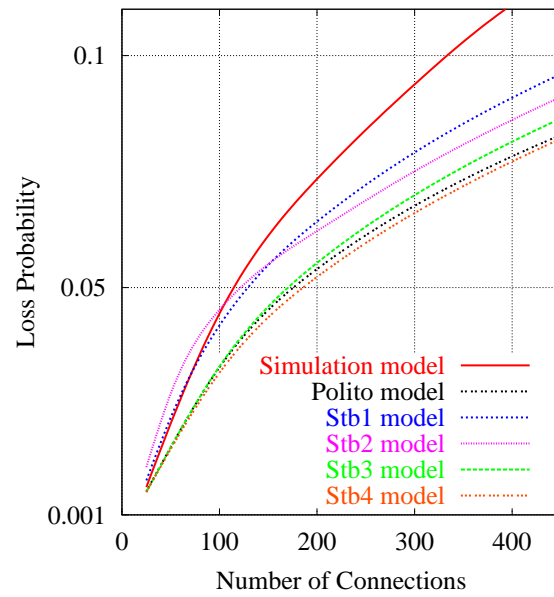
Buffer Formula	TCP tick (ms)	Buffer (packets)	Best Model
$M/M/1/K$	500	128	<i>Stb1</i>
		512	<i>Stb1</i>
	100	128	<i>Stb3</i>
		512	<i>Stb2</i>
Geometric	500	128	<i>Stb1</i>
		512	<i>Stb1</i>
	100	128	<i>Stb4</i>
		512	<i>Stb3</i>

Table 7.1: Single bottleneck loss probability comparison of the models

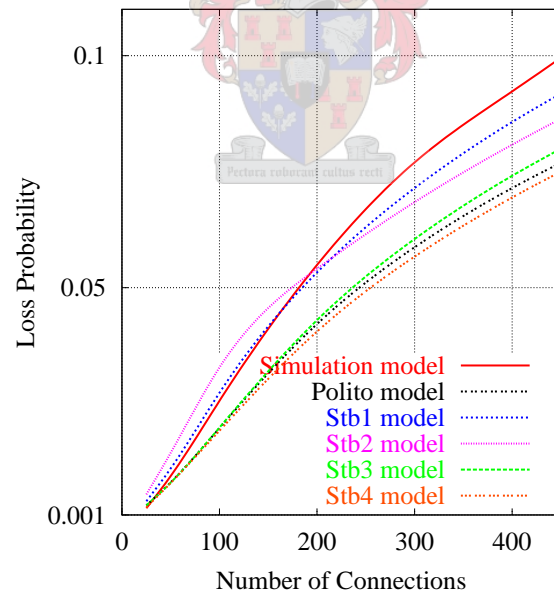
In the literature, the Polito models, [38, 39] and other models such as [1, 40, 66] use a logarithmic scale to present the loss probabilities. Presenting the loss probability results using a log-scale de-emphasizes the differences between the analytical and the *ns2* results. This is illustrated in Figure 7.2(a) which is in linear scale as Figure 7.3 which is in log-scale.

7.4 Average *cwnd* and *ssthresh*

Figures 7.7 through 7.10 present the average *cwnd* size and *ssthresh* values obtained using the *Stb1*, *Stb2*, *Stb3* and *Stb4* models as a function of the number of active TCP connections when the $M/M/1/K$ and geometric buffer formulas are used on the MI-NY link. TCP tick values of 100ms and 500ms and buffer capacities of 128 and 512 packets are used. The comparison with *ns2* simulations shows that our models give accurate average *cwnd* and *ssthresh* values.



(a) Packet loss probability: the buffer capacity is 128 packets



(b) Packet loss probability: the buffer capacity is 512 packets

Figure 7.2: The packet loss probability when a TCP tick value of 500ms and the $M/M/1/K$ buffer formula are used

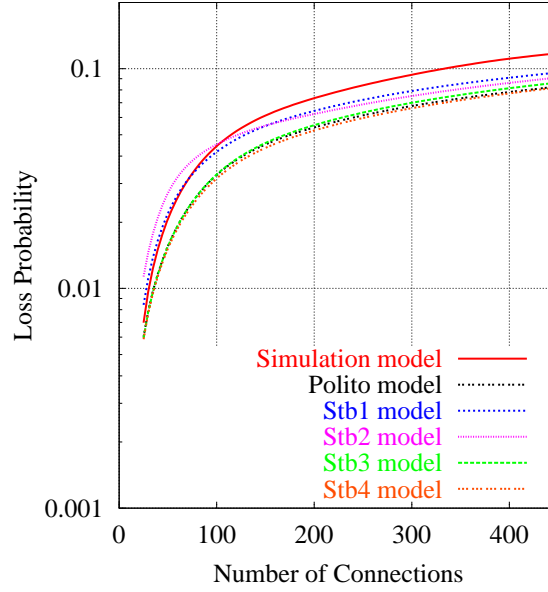


Figure 7.3: The packet loss probability (log-scale) when the TCP tick value is 500ms, the buffer capacity is 128 packets and the $M/M/1/K$ buffer formula is used

7.5 The *cwnd* size distribution

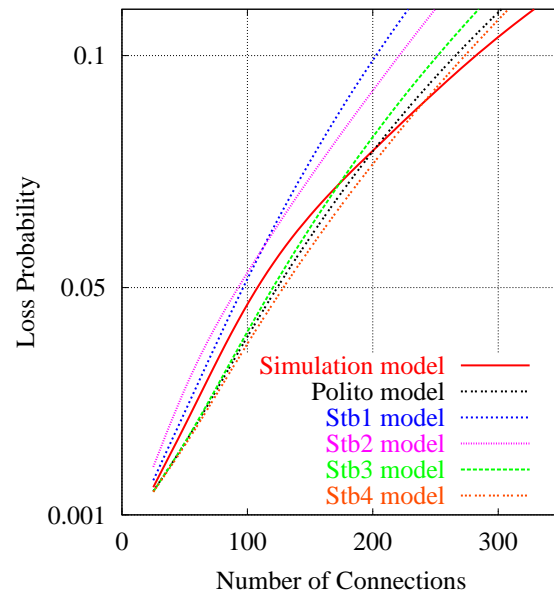
The probability that the *cwnd* size is i in the models of this chapter is calculated by

$$P(cwnd = i) = \frac{\sum_{A \in \mathcal{T}} N(A_i)}{\sum_{A \in \mathcal{T}} \sum_{j=a_{min}}^{a_{max}} N(A_j)}$$

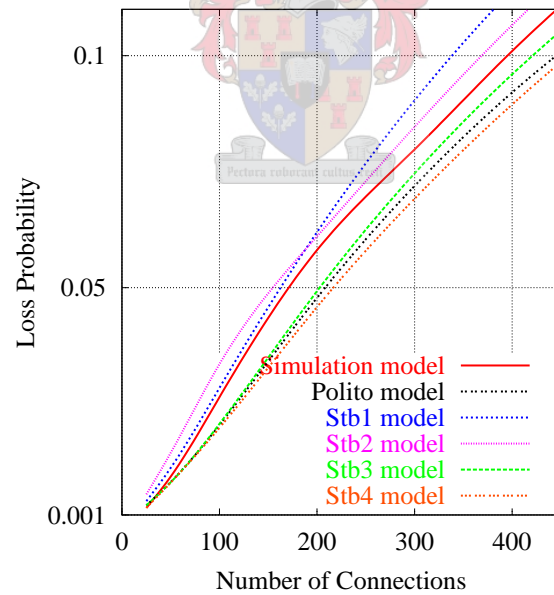
where as given in section 6.1.6 $N(A_i)$ is the number of active TCP connections in state A_i . The a_{min} and a_{max} are the minimum and maximum *cwnd* sizes in state A .

Figures 7.11 through 7.15 present the *cwnd* size distribution when the number of active TCP connections is 25, 100 and 400 respectively. In all cases the *Stb4* model gives an accurate *cwnd* size distribution and the *Stb1* and *Stb2* models underestimate the probability that the *cwnd* size is one. When the number of active TCP connections is 25 the *Stb2* model also underestimates the probabilities that the *cwnd* is two, three and four. The reasons for these results are the different behaviors of the truncated and bounded geometric distributions discussed in sections 6.1.2 and 6.1.2.

Besides when the number of active TCP connections is 25 the *Stb2* model overestimates the probabilities that the *cwnd* size is between 12 and 26. The reason for this is the different behaviors of the *SS1* and *CA2* models of section 6.1.2 from which the *Stb2* model is formed. This can be verified from Figure 7.11 which shows that the *Stb1* model formed from *SS1*

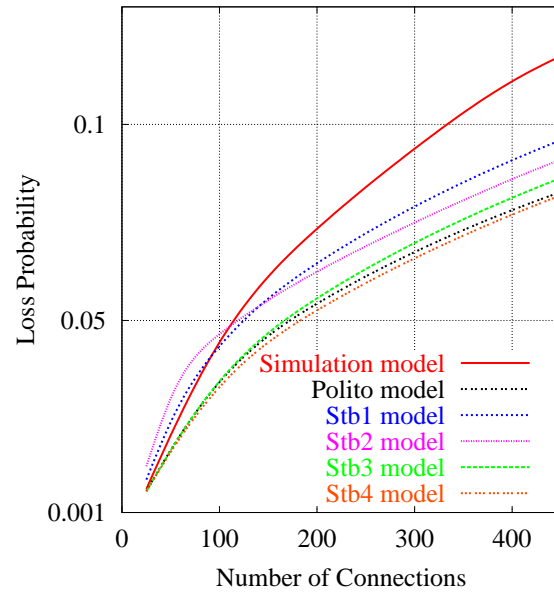


(a) Packet loss probability: the buffer capacity is 128 packets

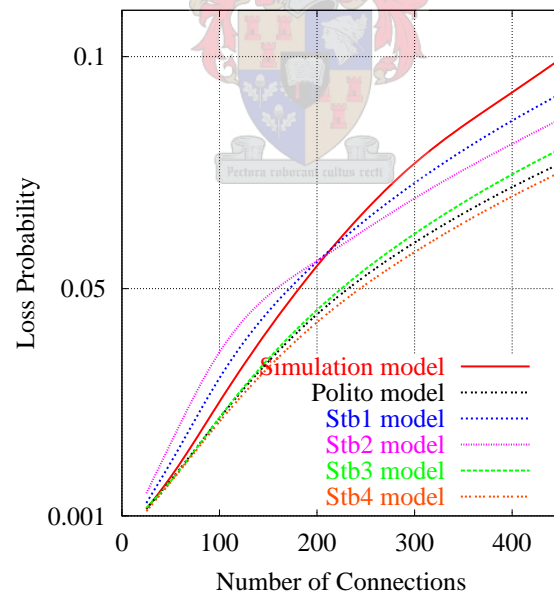


(b) Packet loss probability: the buffer capacity is 512 packets

Figure 7.4: The packet loss probability when a TCP tick value of 100ms and the $M/M/1/K$ buffer formula are used

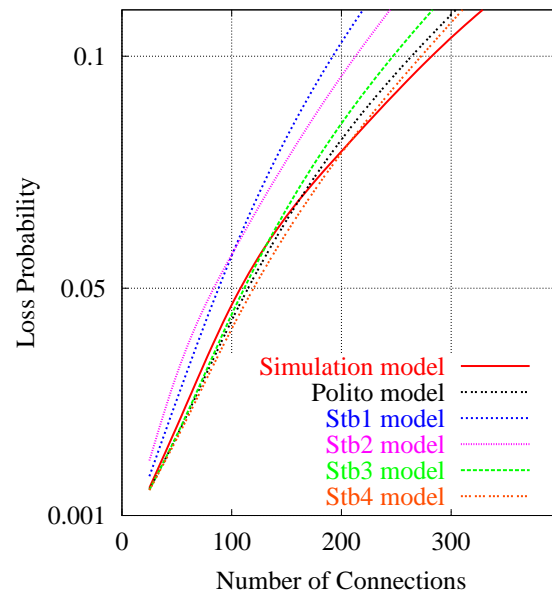


(a) Packet loss probability: the buffer capacity is 128 packets

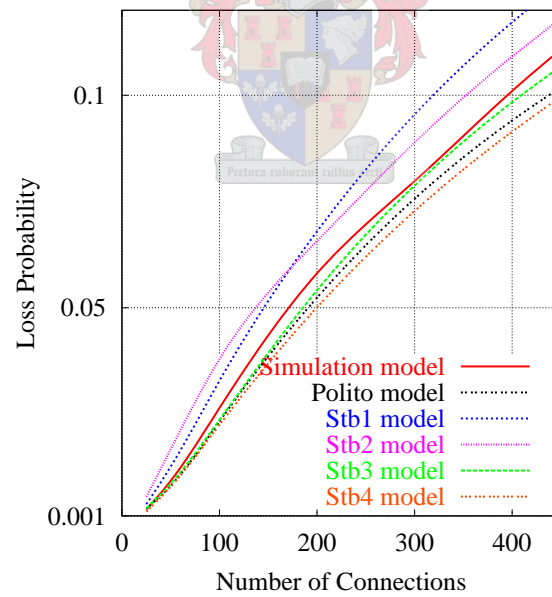


(b) Packet loss probability: the buffer capacity is 512 packets

Figure 7.5: The packet loss probability when a TCP tick value of 500ms and geometric buffer formula are used

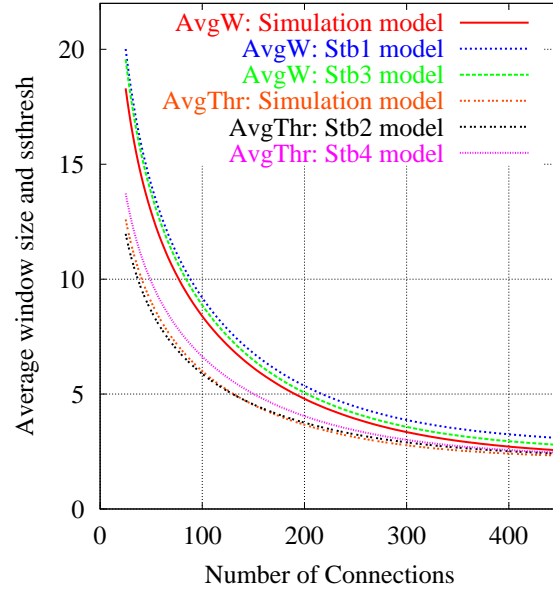


(a) Packet loss probability: the buffer capacity is 128 packets

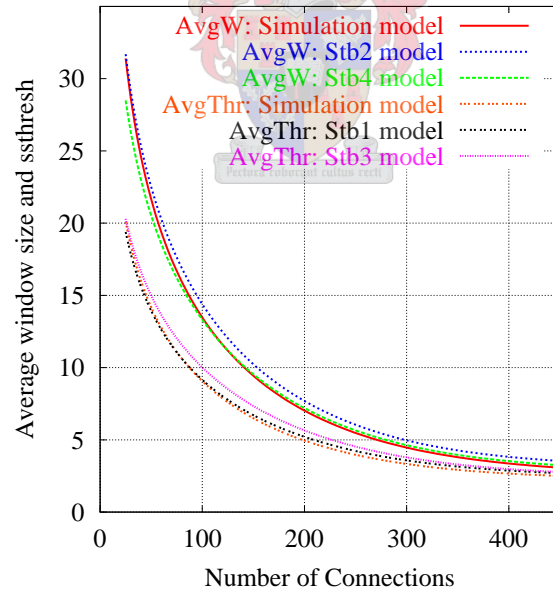


(b) Packet loss probability: the buffer capacity is 512 packets

Figure 7.6: The packet loss probability when a TCP tick value of 100ms and geometric buffer formula are used

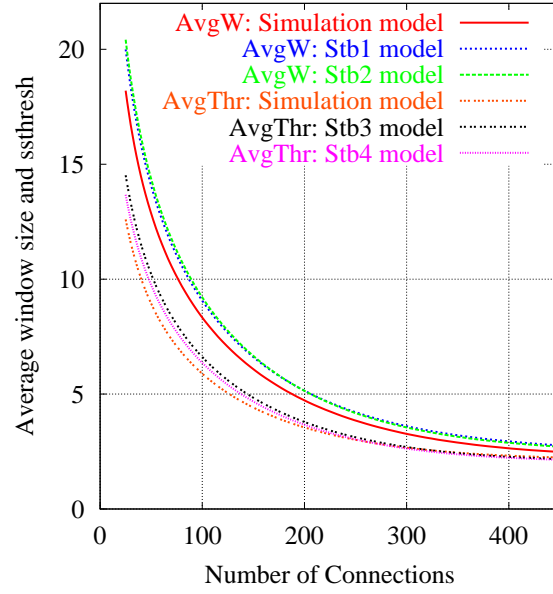


(a) Average *cwnd* size and *ssthresh*: the buffer capacity is 128 packets

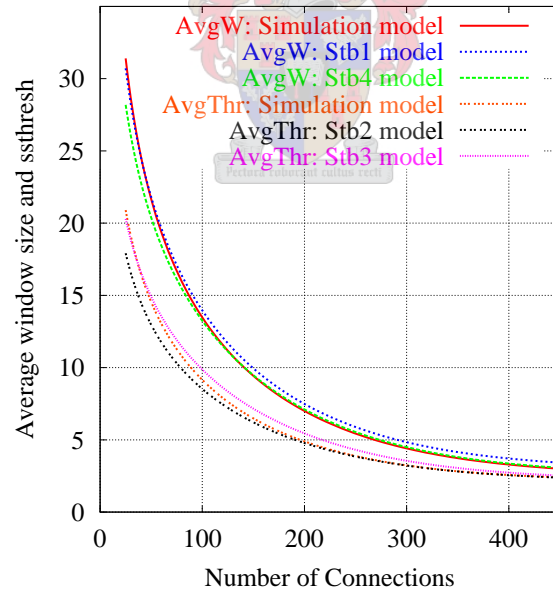


(b) Average *cwnd* size and *ssthresh*: the buffer capacity is 512 packets

Figure 7.7: Average *cwnd* size and *ssthresh* when a TCP tick value of 500ms and the $M/M/1/K$ buffer formula are used

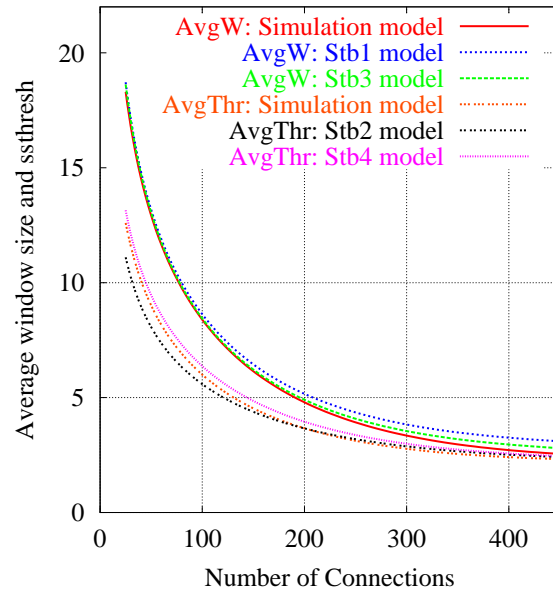


(a) Average *cwnd* size and *ssthresh*: the buffer capacity is 128 packets

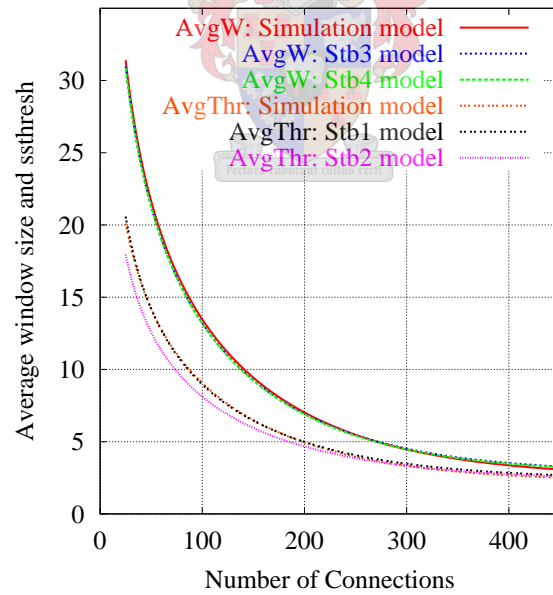


(b) Average *cwnd* size and *ssthresh*: the buffer capacity is 512 packets

Figure 7.8: Average *cwnd* size and *ssthresh* when a TCP tick value of 100ms and the $M/M/1/K$ buffer formula are used

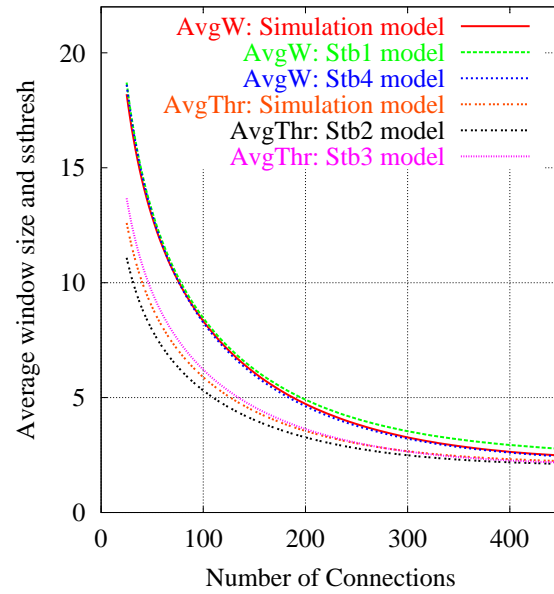


(a) Average *cwnd* size and *ssthresh*: the buffer capacity is 128 packets

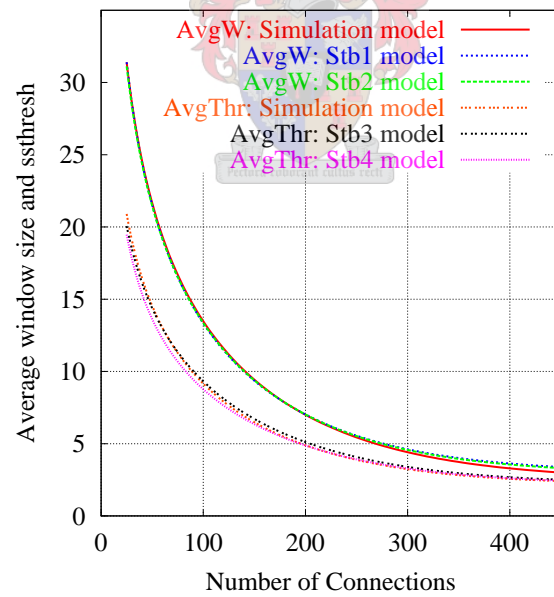


(b) Average *cwnd* size and *ssthresh*: the buffer capacity is 512 packets

Figure 7.9: Average *cwnd* size and *ssthresh* when a TCP tick value of 500ms and geometric buffer formula are used



(a) Average *cwnd* size and *ssthresh*: the buffer capacity is 128 packets



(b) Average *cwnd* size and *ssthresh*: the buffer capacity is 512 packets

Figure 7.10: Average *cwnd* size when a TCP tick value of 100ms and geometric buffer formula are used

and *CA1* slightly overestimates the probabilities that the *cwnd* is between 6 and 16. This overestimation is due to the behavior of the *SS1* model. The *CA1* model (see section 6.1.2) does not consider transitions from *SS* and hence underestimates the probabilities that the *cwnd* values are less than $W_m/2$. The fact that the *CA2* model is based on the *SS1* model further increases the probabilities that the *cwnd* is between 12 and 26. When the number of active TCP connections is 25 the *Stb3* model formed from the *SS2* and *CA1* underestimates the probabilities that the *cwnd* size is between 12 and 26. The reason for this is that the *CA1* model underestimates the probabilities that the *cwnd* values are less than $W_m/2$. It should be noted as shown in Table 7.1 that the *Stb1* and *Stb2* models give more accurate loss probabilities than *Stb4* indicating that the *Stb1* and *Stb2* models may capture the TCP burstiness (see [46, 90]) more than the *Stb4* model.

As can be seen from the figures there is little difference between using the $M/M/1/K$ and geometric buffer formulas when computing the *cwnd* size distribution. The standard TCP tick value of 500 ms and a buffer capacity of 128 packets are used in these experiments. A summary of the model behaviors with regard to the *cwnd* size distribution is given in Table 7.2.

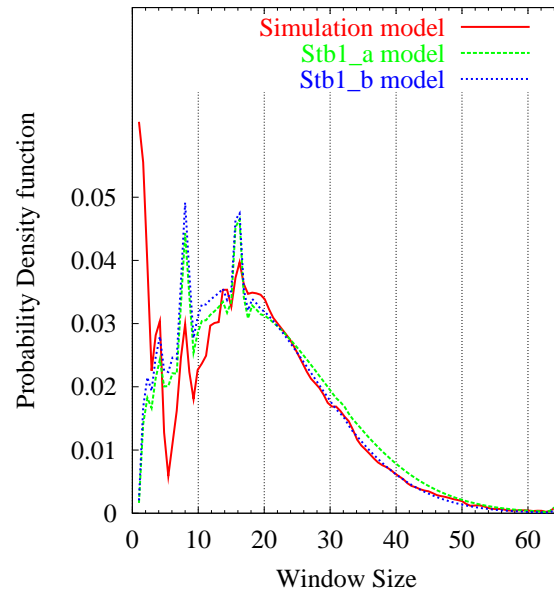
As can be seen from the *cwnd* size distributions, 2^i , $1 \leq i \leq 5$ is more likely than the other *cwnd* values. To account for these higher probabilities of *cwnd* values a factor $\sigma \in [1/w, 1]$ is used in the calculation of the times TCP spends in the SS_i states as shown in Appendix C. The smoothed *cwnd* size distributions shown in Figures 7.13(a) and 7.13(b) use $\sigma = 1/w$ and the non-smoothed *cwnd* sized distributions use $\sigma = 2/3$.

7.6 The *ssthresh* distribution

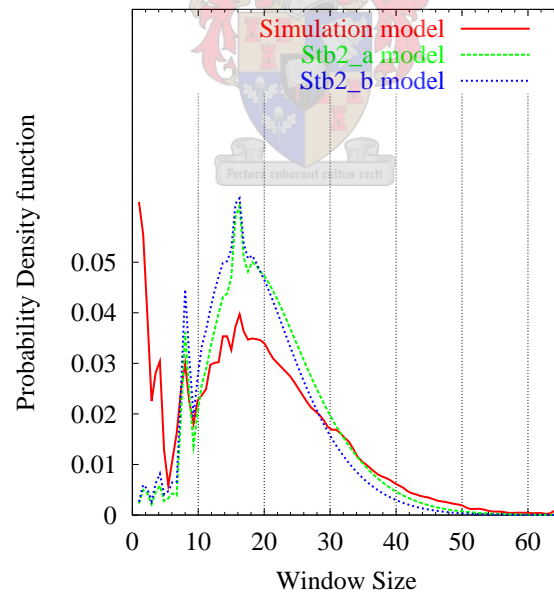
Figures 7.16 through 7.19 present the TCP *ssthresh* distribution when the number of active TCP connections is 25, 100 and 400 respectively. When the number of active TCP connections is 25, the *Stb2* model gives the best *ssthresh* distribution and the *Stb3* model underestimates some of the *ssthresh* values. As explained in section 7.5 above, the *CA1* model underestimates the *cwnd* values less than $W_m/2$. This is the reason why the *Stb3* model also underestimates some *ssthresh* values. When the number of active TCP connections is 100, the *Stb2* model slightly overestimates the probability that the *ssthresh* is 4 and gives a good match with the simulation for all other *ssthresh* values. All other models give accurate *ssthresh* distribution when the number of active TCP connections is 100. When the number of active TCP connections is 400 all models give accurate estimations of the *ssthresh* distribution.

Number of active TCP Connections	Model	Model Behavior
25	<i>Stb1</i>	Underestimates the probability that the <i>cwnd</i> is 1 and gives an accurate prediction of other <i>cwnd</i> values.
	<i>Stb2</i>	Underestimates probabilities that the <i>cwnd</i> is 1, 2, 3, 4 , overestimates the probabilities that <i>cwnd</i> is between 14 and 22 and gives good predictions of other <i>cwnd</i> values.
	<i>Stb3</i>	Underestimates the probabilities that the <i>cwnd</i> is about 15 and 17 and gives accurate predictions of other <i>cwnd</i> values.
	<i>Stb4</i>	Gives accurate predictions of all <i>cwnd</i> values.
100	<i>Stb1</i>	Underestimates the probability that the <i>cwnd</i> is 1 and gives accurate predictions of other <i>cwnd</i> values.
	<i>Stb2</i>	Underestimates the probability that the <i>cwnd</i> is 1 and gives accurate predictions of other <i>cwnd</i> values.
	<i>Stb3</i>	Gives accurate predictions of all <i>cwnd</i> values.
	<i>Stb4</i>	Gives accurate predictions of all <i>cwnd</i> values.
400	<i>Stb1</i>	Underestimates the probability that the <i>cwnd</i> is 1 and gives accurate predictions of other <i>cwnd</i> values.
	<i>Stb2</i>	Underestimates the probability that the <i>cwnd</i> is 1 and gives accurate predictions of other <i>cwnd</i> values.
	<i>Stb3</i>	Gives accurate predictions of all <i>cwnd</i> values.
	<i>Stb4</i>	Gives accurate predictions of all <i>cwnd</i> values.

Table 7.2: Comparison of the *cwnd* size distributions

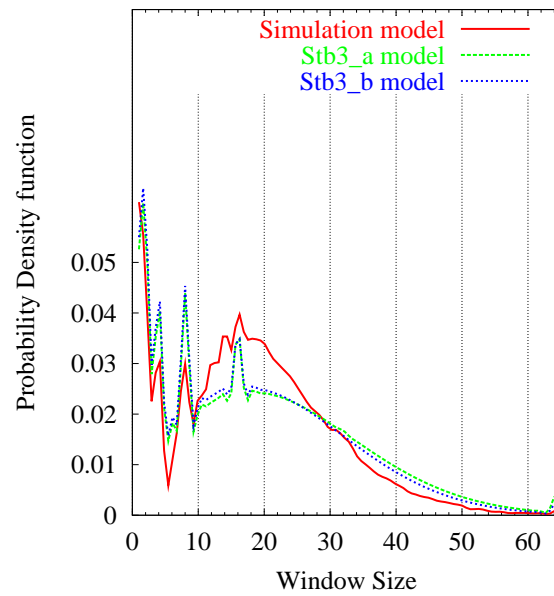


(a) The $cwnd$ size distribution using the $Stb1$ model with the $M/M/1/K$ buffer formula ($Stb1_a$) and with the geometric buffer formula ($Stb1_b$)

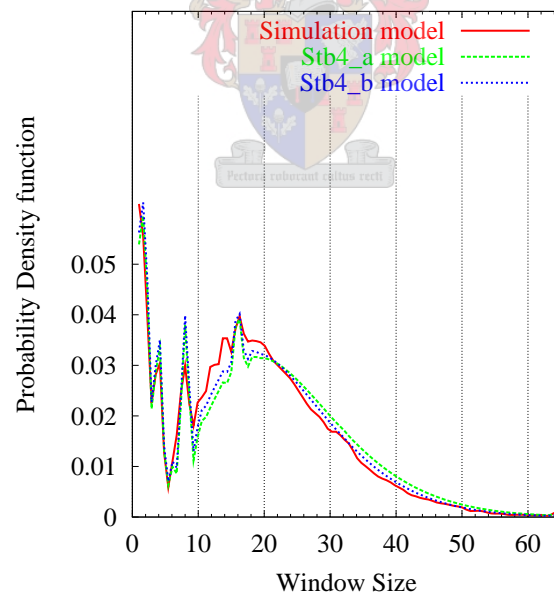


(b) The $cwnd$ size distribution using the $Stb2$ model with the $M/M/1/K$ buffer formula ($Stb2_a$) and with the geometric buffer formula ($Stb2_b$)

Figure 7.11: The $cwnd$ size distribution for 25 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms

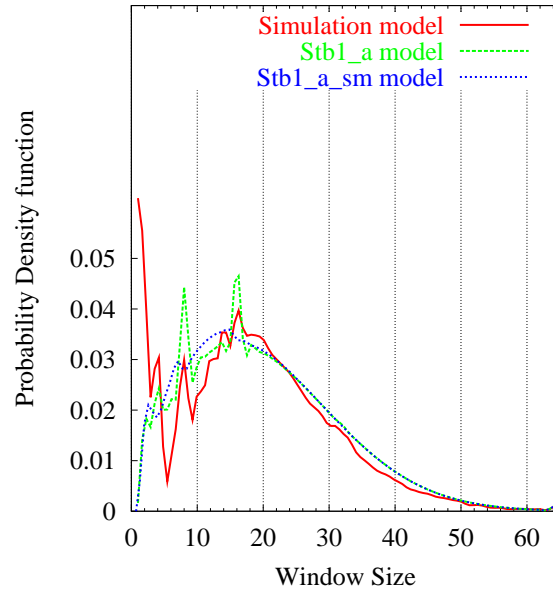


(a) The *cwnd* size distribution using the *Stb3* model with the $M/M/1/K$ buffer formula (*Stb3_a*) and with the geometric buffer formula (*Stb3_b*)

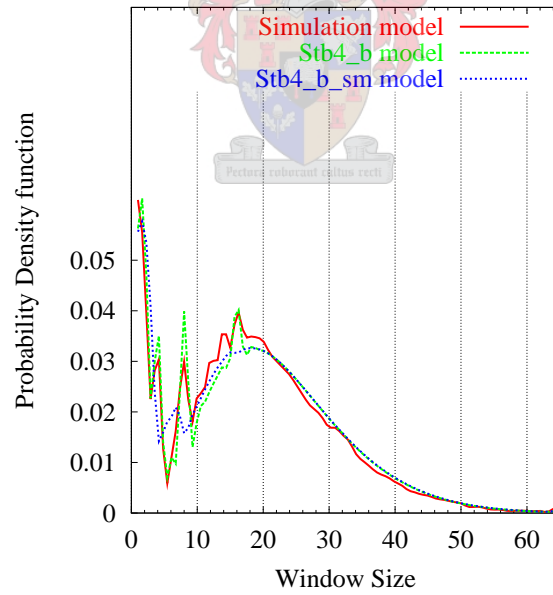


(b) The *cwnd* size distribution using the *Stb4* model with the $M/M/1/K$ buffer formula (*Stb4_a*) and with the geometric buffer formula (*Stb4_b*)

Figure 7.12: The *cwnd* size distribution for 25 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms

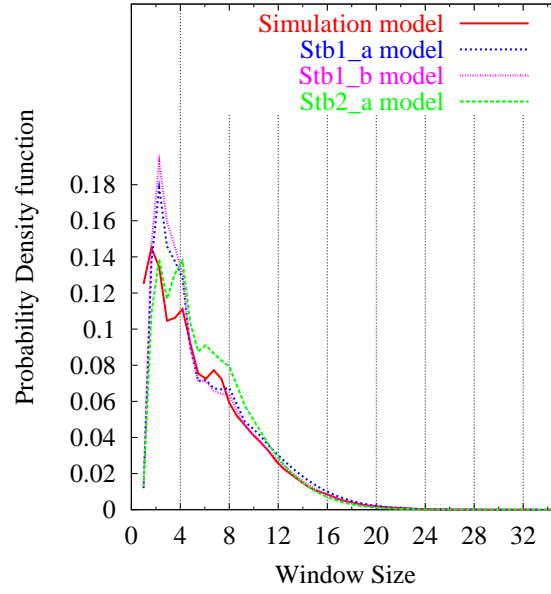


(a) smoothed versus non-smoothed *cwnd* size distributions using the *Stb1* model with the $M/M/1/K$ buffer formula (Stb1_a)

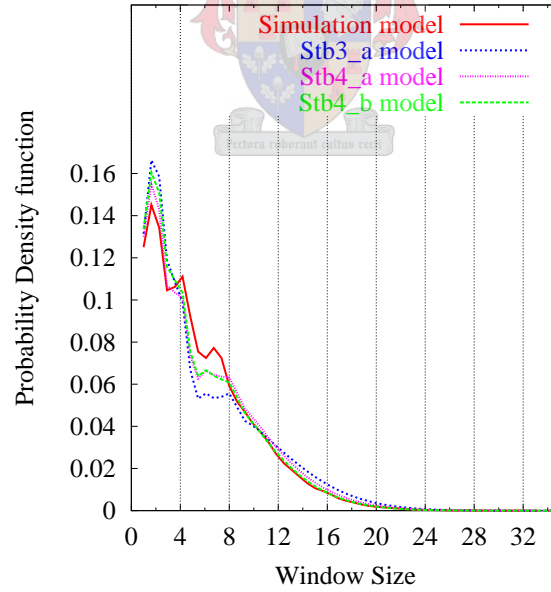


(b) Smoothed and non-smoothed *cwnd* size distributions using the *Stb4* model with the $M/M/1/K$ buffer formula (Stb4_a)

Figure 7.13: smoothed and non-smoothed *cwnd* size distributions for 25 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms

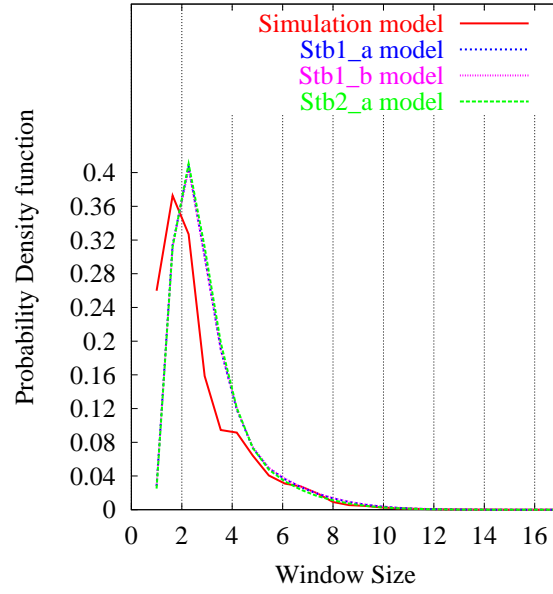


(a) The *cwnd* size distribution using *Stb1* with the $M/M/1/K$ buffer formula (*Stb1_a*) and with the geometric buffer formula (*Stb1_b*) and the *Stb2* model with the $M/M/1/K$ buffer formula (*Stb2_a*)

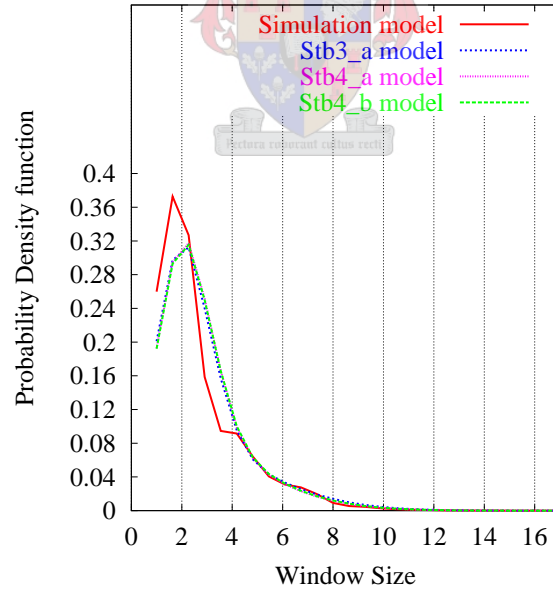


(b) The *cwnd* size distribution using *Stb4* with the $M/M/1/K$ buffer formula (*Stb4_a*) and with the geometric buffer formula (*Stb4_b*) and the *Stb3* model with the $M/M/1/K$ buffer formula (*Stb3_a*)

Figure 7.14: The *cwnd* size distribution when the buffer capacity is 128 packets and the TCP tick value is 500 ms for 100 active TCP connections

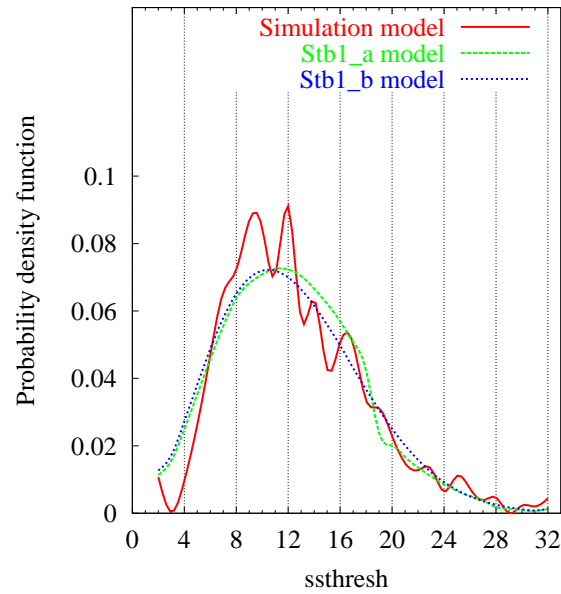


(a) The *cwnd* size distribution using *Stb1* with the $M/M/1/K$ buffer formula (*Stb1_a*) and with the geometric buffer formula (*Stb1_b*) and the *Stb2* model with the $M/M/1/K$ buffer formula (*Stb2_a*)

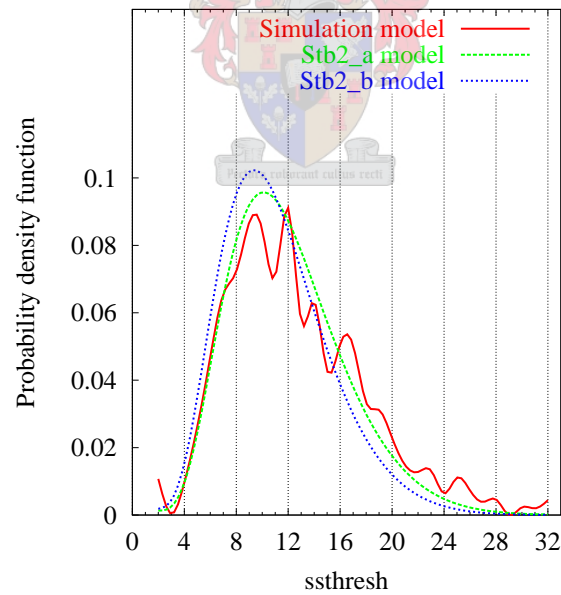


(b) The *cwnd* size distribution using *Stb4* with the $M/M/1/K$ buffer formula (*Stb4_a*) and with the geometric buffer formula (*Stb4_b*) and the *Stb3* model with the $M/M/1/K$ buffer formula (*Stb3_a*)

Figure 7.15: The *cwnd* size distribution when the buffer capacity is 128 packets and the TCP tick value is 500 ms for 400 active TCP connections

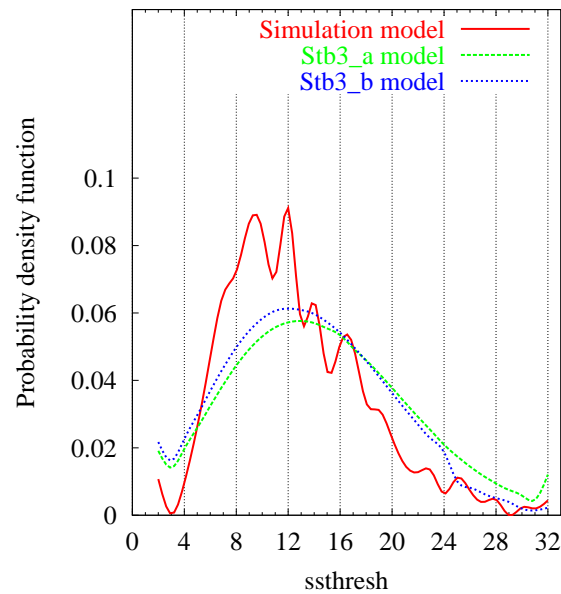


(a) The *ssthresh* distribution using the *Stb1* model with the $M/M/1/K$ buffer formula (*Stb1_a*) and with the geometric buffer formula (*Stb1_b*)

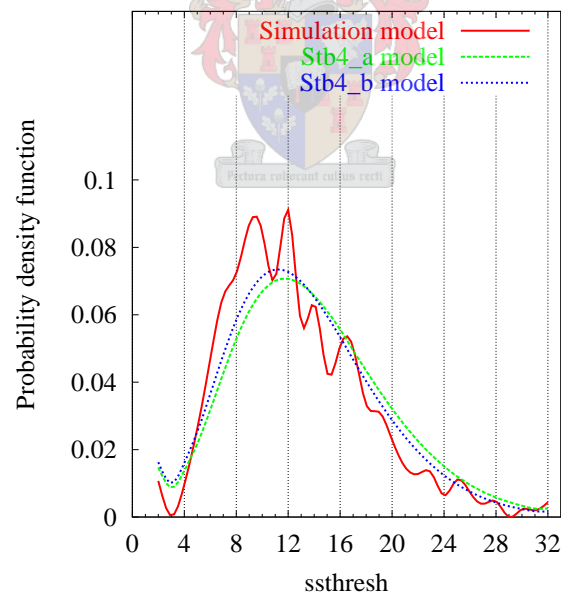


(b) The *ssthresh* distribution using the *Stb2* model with the $M/M/1/K$ buffer formula (*Stb2_a*) and with the geometric buffer formula (*Stb2_b*)

Figure 7.16: The *ssthresh* distribution for 25 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms

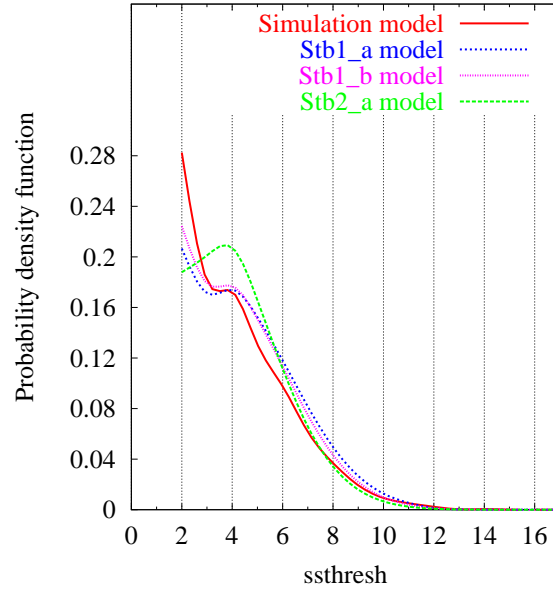


(a) The *ssthresh* distribution using the *Stb3* model with the $M/M/1/K$ buffer formula (*Stb3_a*) and with the geometric buffer formula (*Stb3_b*)

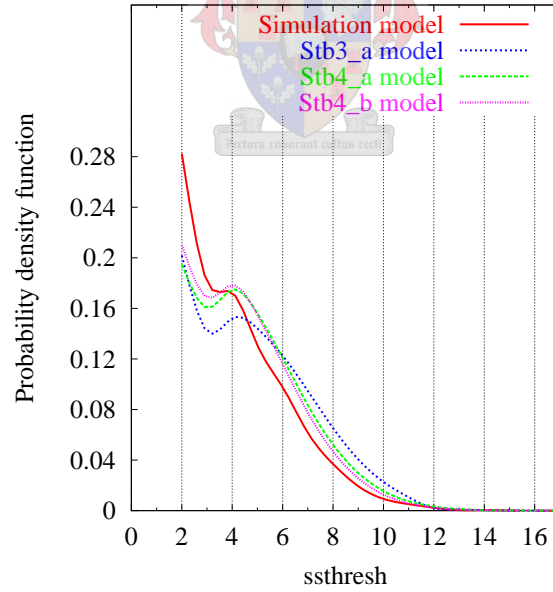


(b) The *ssthresh* distribution using the *Stb4* model with the $M/M/1/K$ buffer formula (*Stb4_a*) and with the geometric buffer formula (*Stb4_b*)

Figure 7.17: The *ssthresh* distribution for 25 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms

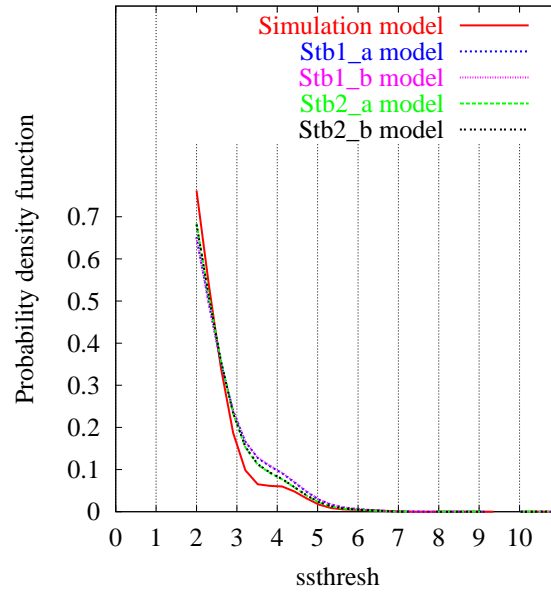


(a) The *ssthresh* distribution using *Stb1* with the $M/M/1/K$ buffer formula (*Stb1_a*) and with the geometric buffer formula (*Stb1_b*) and the *Stb2* model with the $M/M/1/K$ buffer formula (*Stb2_a*)

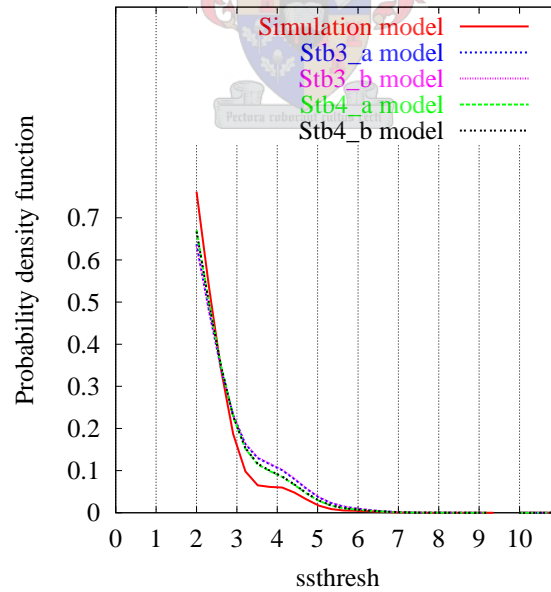


(b) The *ssthresh* distribution using *Stb4* with the $M/M/1/K$ buffer formula (*Stb4_a*) and with the geometric buffer formula (*Stb4_b*) and the *Stb3* model with the $M/M/1/K$ buffer formula (*Stb3_a*)

Figure 7.18: The *ssthresh* distribution for 100 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms



(a) The *ssthresh* distribution using *Stb1* with the $M/M/1/K$ buffer formula (*Stb1_a*) and with the geometric buffer formula (*Stb1_b*) and the *Stb2* model with the $M/M/1/K$ buffer formula (*Stb2_a*) and with the geometric buffer formula (*Stb2_b*)



(b) The *ssthresh* distribution using *Stb4* with the $M/M/1/K$ buffer formula (*Stb4_a*) and with the geometric buffer formula (*Stb4_b*) and the *Stb3* model with the $M/M/1/K$ buffer formula (*Stb3_a*) and with the geometric buffer formula (*Stb3_b*)

Figure 7.19: The *ssthresh* distribution for 400 active TCP connections when the buffer capacity is 128 packets and the TCP tick value is 500 ms

7.7 The timeout probability

Our models give some performance metrics which may be difficult to obtain using *ns2* simulations. For example the fraction of time that TCP connections wait for a timeout to expire and the average *cwnd* excluding the window size while TCP connections wait for a timeout to expire as a function of the number of active TCP connections can be computed using our models.

The probability that TCP waits for a timeout to expire is given by

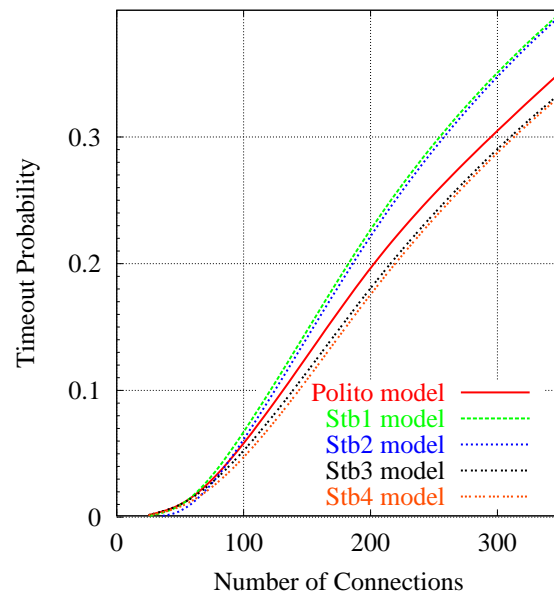
$$P(\text{timeout}) = \frac{\sum_{i=1}^{W_m/2} N(SST_i) + \sum_{i=2}^{W_m} N(CAT_i)}{\sum_{A \in \mathcal{T}} \sum_{i=a_{min}}^{a_{max}} N(A_i)}$$

where as given in section 6.1.6 $N(A_i)$ is the number of active TCP connections in state A_i . The a_{min} and a_{max} are the minimum and maximum *cwnd* sizes in states A and are given in Table 6.1.

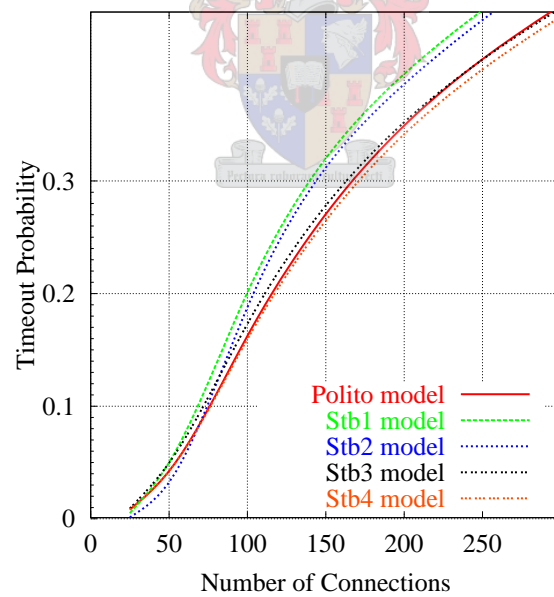
Figures 7.20, 7.21 and 7.22(a) present the probability that TCP waits for timeout to expire for each of our models against the Polito model. The figures show that the timeout probability depends upon the number of active TCP connections. From Figure 7.22(a) it can also be seen that the timeout probability is higher when a smaller buffer capacity (128 packets) and the standard TCP tick value (500 ms) are used. This is because if the buffer capacity is smaller then the loss probability (timeout loss) is higher. Besides if the TCP tick value is higher then we have a bigger timeout value as shown in equation 2.3 and the probability that TCP waits for timeout to expire becomes larger.

The average *cwnd* size including and excluding the *cwnd* size values of TCP connections experiencing timeout as a function of the number of connections is given in Figure 7.22(b). From the figure it can be seen that when the number of active TCP connections is high, the average *cwnd* size excluding the window values of TCP connections experiencing timeout is almost half of the *cwnd* sizes including the window values of TCP connections experiencing timeout.

As explained in [39] in TCP implementations and the *ns2* simulator the window value for TCP connections experiencing a timeout is the window value which the connections had before the timeout. However no packet is transmitted during this timeout period. Therefore the average traffic offered by a TCP connection to the network cannot be obtained as the ratio of the average window size including the timed-out connections to the *RTT*. It can however be estimated by the ratio of the average window size excluding the timed-out connections to the *RTT*.

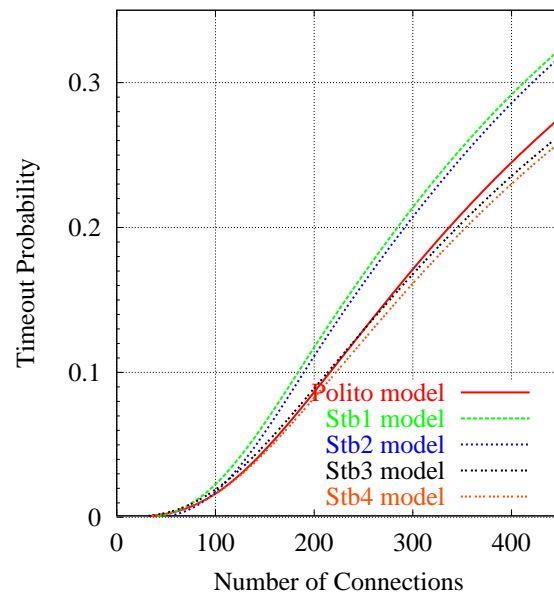


(a) The fraction of the time TCP waits for timeout to expire: the TCP tick value is 100 ms and the $M/M/1/K$ buffer formula is used

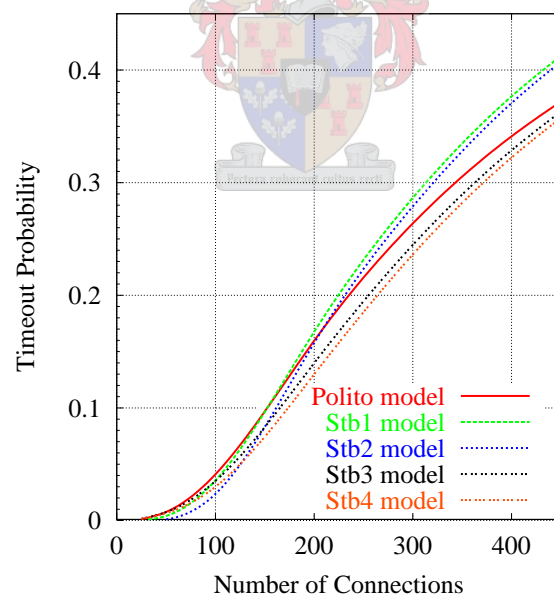


(b) The fraction of the time TCP waits for timeout to expire: the TCP tick value is 500 ms and the geometric buffer formula is used

Figure 7.20: Fraction of time TCP waits for timeout to expire when the buffer capacity is 128 packets



(a) The fraction of the time TCP waits for timeout to expire: the TCP tick value is 100 ms and the geometric buffer formula is used



(b) The fraction of the time TCP waits for timeout to expire: the TCP tick value is 500 ms and the $M/M/1/K$ buffer formula is used

Figure 7.21: Fraction of time TCP waits for timeout to expire when the buffer capacity is 512 packets

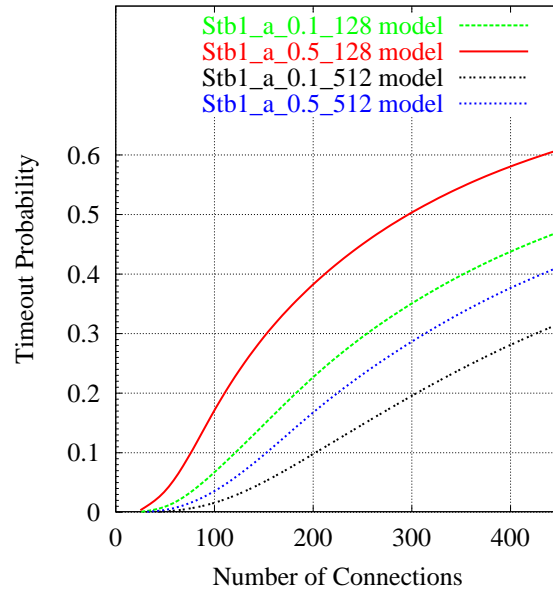
7.8 The buffer occupancy

Figures 7.23 and 7.24 present the TCP buffer occupancy for *Stb1*, *Stb2*, *Stb3* and *Stb4* when the $M/M/1/K$ and geometric buffer formulas are used. In all figures the $M/M/1/K$ buffer formula gives more accurate values of the buffer occupancy than the geometric buffer formula. Using the $M/M/1/K$ buffer formula, the *Stb3* and *Stb4* models give the most accurate predictions of the buffer occupancy. The results obtained from the geometric buffer formula have a similar trend as the results obtained from the *ns2* simulation. The range of values where all of the models overestimate or underestimate the buffer occupancy when the $M/M/1/K$ and geometric buffer formulas are used as a function of the number of active TCP connections n is given in Table 7.3.

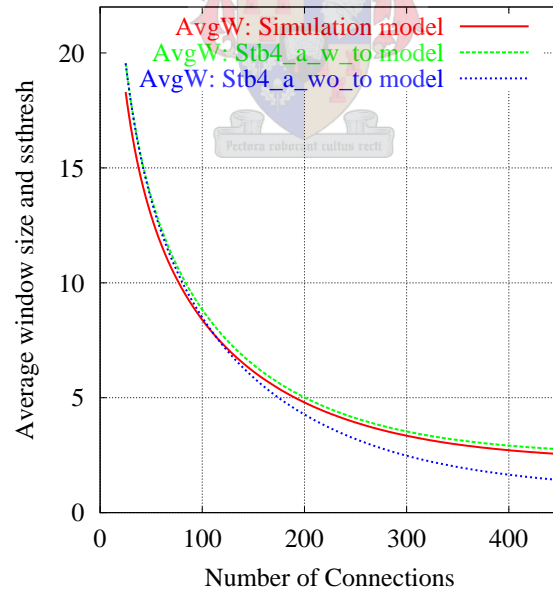
TCP tick (ms)	Buffer (packets)	$M/M/1/K$	Geometric
100	128	Overestimates for $n < 200$	Underestimates for $n < 300$ and overestimates for $n > 300$
	512	Underestimates for $n > 250$ and overestimates when $n < 150$	Underestimates for all values of n
500	128	Overestimates for $n < 200$	Underestimates for $n < 250$ and overestimates for $n > 250$
	512	Underestimates for $n > 300$ and Overestimates when $n < 200$	Underestimates for all values of n

Table 7.3: Buffer occupancy comparison of the $M/M/1/K$ and geometric buffer formulas

Taking the average of the buffer values obtained using $M/M/1/K$ and geometric buffer formulas, a good match can be obtained with the simulation.



(a) Comparisons of the fractions of the time TCP waits for timeout to expire when the buffer capacities are 128 and 512 packets, the TCP tick values are 100 and 500 ms and the $M/M/1/K$ buffer formula is used



(b) Comparison of the average *cwnd* size including (Stb4_a_w_to) and excluding (Stb4_a_wo_to) the *cwnd* values where TCP waits for timeout to expire for the *Stb4* model when the $M/M/1/K$ buffer formula is used (Stb4_a)

Figure 7.22: Comparisons of some performance metrics which are not easy to obtain using *ns2* simulations

7.9 Analysis of Individual Connections

Figures 7.25 and 7.26, show the throughput of individual connections when the buffer capacity is 512 packets and TCP tick values of 100 ms and 500 ms are used. These results are when both the $M/M/1/K$ and the geometric buffer formulas are used. In the figures the letter “a” is used to indicate the $M/M/1/K$ and the letter “b” is used to indicate the geometric buffer formula. The decimals 0.5 and 0.1 next to the letters denote the TCP tick value in seconds. The integers 128 and 512 next to the tick values denote the buffer capacities in packets. From the figures it can be seen that all models give similar predictions of the throughput. When the buffer capacity is 128 packets (see Figure 7.26), the coefficient of variation of the throughput values is high.

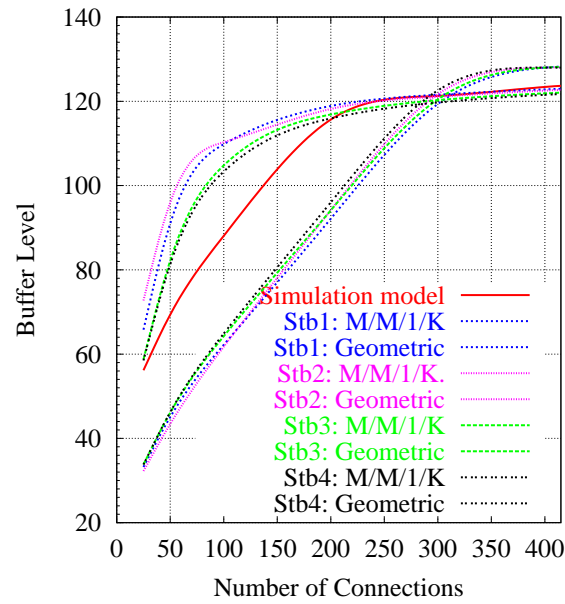
The figures show that our models slightly underestimate the throughput. This is due to the assumption of an equal average loss probability for all connections. If the exact packet loss probability can be obtained from simulation or measurements, the throughput of the individual connections can be estimated accurately. The TCP sub-models give accurate average *cwnd* size and *ssthresh* values as verified in section 7.4.

From the analytical models it can also be seen that for a buffer capacity of 128 packets the tick value of 500ms gives higher throughput specially when the distance of the connections is long. For short TCP connections a tick value of 100ms seems to give better throughput. However when the router buffer capacity is 512 packets both tick values give a similar throughput. Besides if the buffer capacity is 128 packets the variation between throughput values of different connections is higher than when a bigger buffer capacity of 512 packets is used.

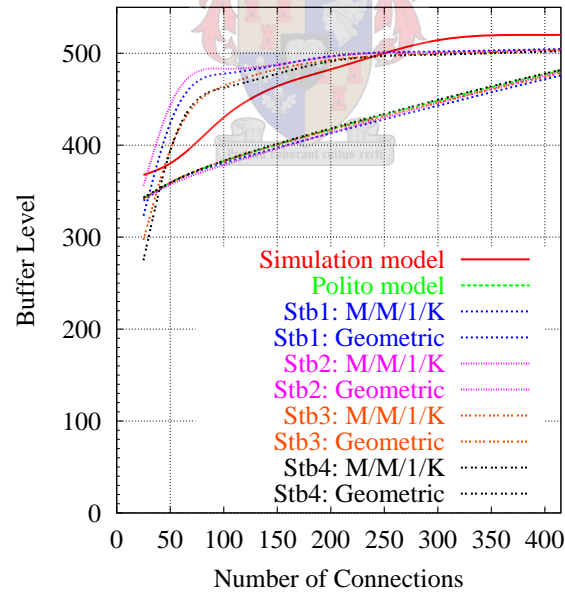
7.10 Chapter summary

This chapter presents a European network topology which is used for the validation of our models. Comparison of the packet loss probability values and *cwnd* size distributions obtained using our *Stb1*, *Stb2*, *Stb3* and *Stb4* models against the Polito model and the *ns2* simulation experiments are given in Tables 7.1 and 7.2 respectively. In all scenarios at least one of our models outperforms the Polito model.

The behavior of the generalized bounded geometric distribution enables our *Stb1* and *Stb2* models to give more accurate predictions of the packet loss probability values than the other models. This behavior of the distribution helps our *Stb1* and *Stb2* models to capture TCP burstiness which causes buffer overflows (packet loss) and queueing delays (see [27]) more than the other

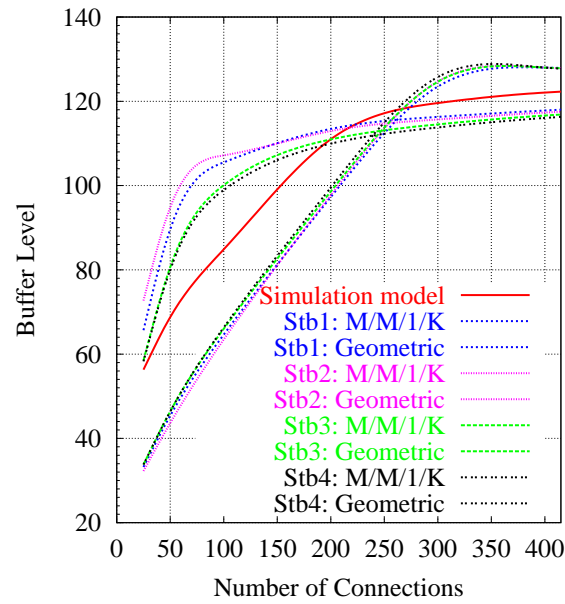


(a) Buffer occupancy: the buffer capacity is 128 packets

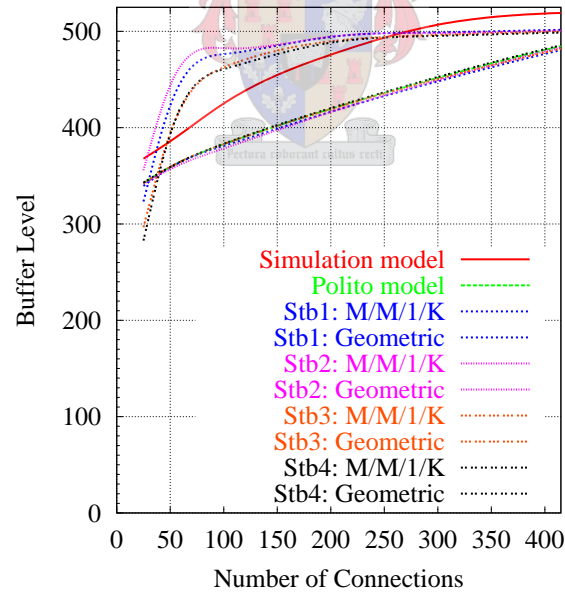


(b) Buffer occupancy: the buffer capacity is 512 packets

Figure 7.23: Buffer occupancy with a TCP tick value of 100 ms

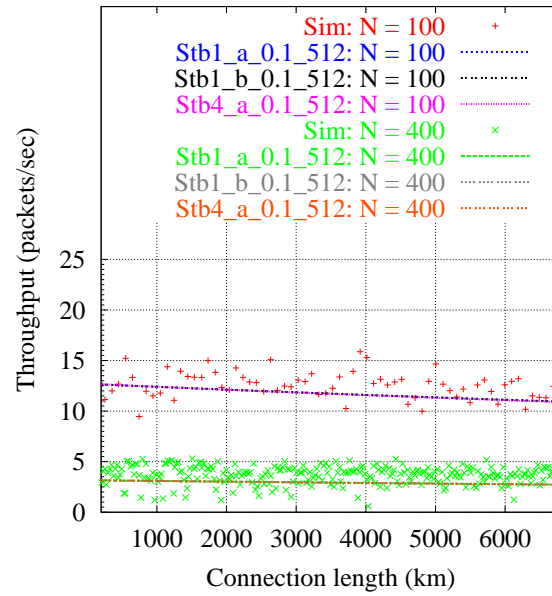


(a) Buffer occupancy: the buffer capacity is 128 packets

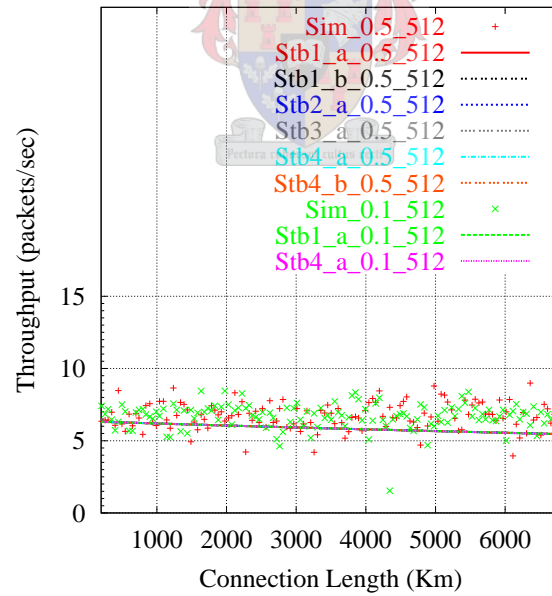


(b) Buffer occupancy: the buffer capacity is 512 packets

Figure 7.24: Buffer occupancy with the standard TCP tick value of 500 ms



(a) Throughput of individual connections: the TCP tick value is 100 ms when the number of active TCP connections is 100 and 400



(b) Throughput of individual connections: the TCP tick value is 100 ms (0.1s) and 500 ms (0.5s) when the number of active TCP connections is 200

Figure 7.25: Throughput of individual connections when the buffer capacity is 512 packets

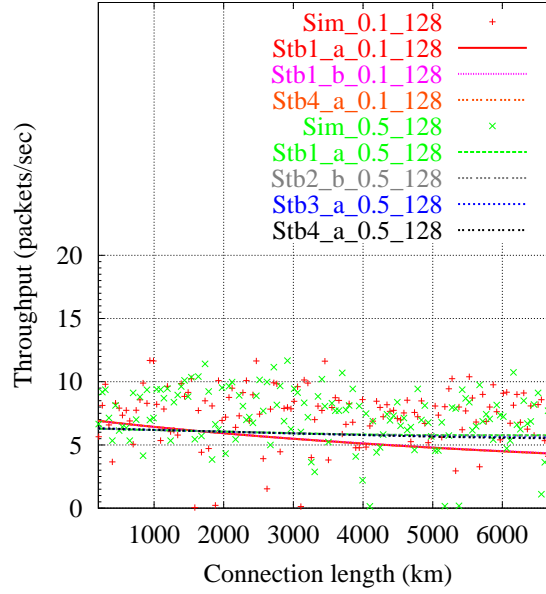


Figure 7.26: Throughput of individual connections: the buffer capacity is 128 packets, the TCP tick value is 100 ms (0.1s) and 500 ms (0.5s) when the number of active TCP connections is 200

models.

The behavior of the generalized truncated geometric distribution and the fact that there is no assumption in the derivation of the *CA2* (see section 6.1.2) model allows our *Stb4* model to give more accurate prediction of the *cwnd* distribution. Our *Stb2* model also gives the best *ssthresh* distribution even though the other models also give good estimates of this distribution.

All models give similar and accurate predictions of the average *cwnd*, *ssthresh* and throughput values.

Our models also give the fraction of time TCP connections wait for a timeout to expire and the average *cwnd* size excluding the window sizes while TCP connections wait for a timeout to expire as a function of the number of active TCP connections. These values may be difficult to obtain using the *ns2* simulation. The average *cwnd* values excluding the window values of TCP connections experiencing TO is almost half of the *cwnd* values including the window values of TCP connections experiencing TO when the number of connections is high. However no packet is transmitted during this period. Therefore the average traffic load offered to the network can only be estimated by the ratio of the average *cwnd* size excluding the timed out connections to the *RTT*.

The model results show that the timeout probability is higher when a smaller router buffer capacity of 128 packets and a higher TCP tick value of 500 ms are used. This is because if the buffer capacity is smaller then the loss probability (timeout loss) is higher. Besides if the TCP tick value is higher, we have longer timeout periods resulting in a higher timeout probability.

In most cases there is no big difference between the results obtained from the $M/M/1/K$ and the geometric techniques. However the $M/M/1/K$ technique gives more accurate estimates of the buffer occupancy (queue length) than the geometric buffer technique. The results obtained from the geometric buffer formula have a similar trend as the simulation results and a combination of these two buffer formulas may give more accurate router buffer occupancy values although not included in this study.

Our models slightly underestimate the throughput of individual TCP connections due to the assumption of an equal packet loss probability for all connections. If exact packet loss probability can be obtained from simulation or measurements, the throughput of the individual connections can be estimated accurately.

Our models show that when the distance of the connections is long a smaller router buffer capacity of 128 packets and the standard tick value of 500 ms result in a higher throughput. When the length of (distance covered by) the connections is short a smaller TCP tick value of 100 ms with the same buffer capacity gives better throughput. It seems that higher tick value is better for long TCP connections and smaller tick value is better for TCP connections which cover a short distance. But when a higher router buffer capacity of 512 packets is used, both tick values give similar results. The variation of the throughput values of different TCP connections sharing the same bottleneck link(s) is higher when the buffer capacity is smaller (128 packets).

Chapter 8

Numerical Results: Multi bottleneck link analysis

This chapter presents the numerical results of multi bottleneck links and is organized as follows. First section 8.1 presents a detailed multi bottleneck analysis of the European network topology shown in Figure 7.1. Then the multi bottleneck loss probability and average *cwnd* results are analyzed in sections 8.2 and 8.3 respectively. We present an analysis of the results of multi bottleneck fat pipes in section 8.4. The chapter summary is given in section 8.5.

8.1 The analysis of multi bottleneck links

This chapter presents numerical results for connections crossing the MI-NY and POLI-TO multi bottleneck links shown in Figure 7.1. Here we are interested in connections for the transfer of Web pages from the USA servers to clients at Politecnico di Torino where both the transoceanic (MI-NY) link and the access link (POLI-TO) are congested. We assume that 25% of the connections traverse both bottlenecks.

The analysis of multi bottleneck links is more complex. For the MI-NY and POLI-TO multi bottleneck topology described in Figure 7.1 we develop the implementation method described in Figure 8.1. This technique can be extended to more complex multi bottleneck network topologies.

Each component of Figure 8.1 is described as follows.

- *TCP sub-model 1* refers to the homogeneous groups (see chapter 6) of TCP connections

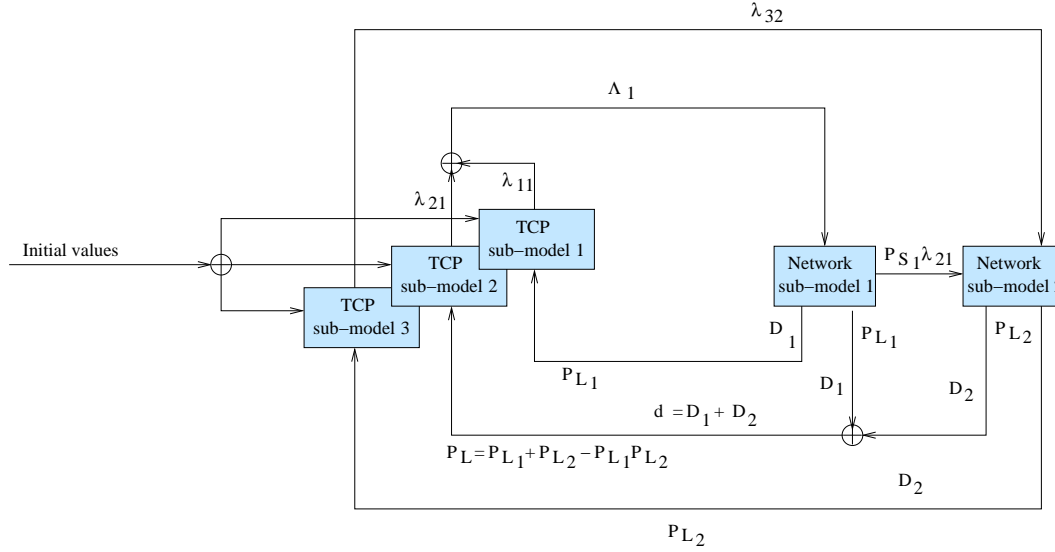


Figure 8.1: The two bottleneck link case: mapping the network topology in Figure 7.1 into sub-models

crossing only the MI-NY link. These are the type 3 connections from USA servers to clients connected to the GARR-B/TEN155 network through the MI-NY link.

- *TCP sub-model 2* refers to the homogeneous groups of TCP connections crossing both the MI-NY and POLI-TO two-bottleneck links. These are the type 1 TCP connections from USA servers to Politecnico di Torino clients.
- *TCP sub-model 3* refers to the homogeneous groups of TCP connections crossing only the POLI-TO link. These are the type 2 TCP connections from GARR-B/TEN155 servers to clients at Politecnico di Torino.
- *Network sub-model 1* refers to the single router at NY.
- *Network sub-model 2* refers to the single router at TO.
- P_{L1} , P_{L2} and P_{S1} denote the packet loss probabilities at the MI-NY and POLI-TO links and the success probability of a packet at the MI-NY links respectively.
- D_1 and D_2 refer to the queuing delays at the MI-NY and POLI-TO links respectively.
- λ_{11} and λ_{21} are the loads offered by *TCP sub-model 1* and *TCP sub-model 2* to the MI-NY link respectively. Λ_1 is the total load offered to the MI-NY link and λ_{32} is the load offered to the POLI-TO link by *TCP sub-model 3*.

We next present the implementation details of the multi bottleneck link model.

8.1.1 Implementation details of the multi bottleneck topology

In the implementation of the multi bottleneck link model first the parameters $P_{L_1}, P_{L_2}, D_1, D_2$ and the *ssthresh* probabilities are initialized. Then the *TCP sub-model 1* which contributes λ_{11} and the *TCP sub-model 2* which contributes λ_{21} both offer $\Lambda_1 = \lambda_{11} + \lambda_{21}$ to *Network sub-model 1*. The *TCP sub-model 3* offers λ_{32} to *Network sub-model 2*. Since the connections represented by *TCP sub-model 2* cross both the MI-NY and POLI-TO bottleneck links, $P_{S_1} \lambda_{21}$ is also offered to *Network sub-model 2* where $P_{S_1} = 1 - P_{L_1}$. Hence the total load offered to bottleneck link 2, the POLI-TO link, is $\Lambda_2 = \lambda_{32} + P_{S_1} \lambda_{21}$.

The *Network sub-model 1* in turn offers P_{L_1} and D_1 to *TCP sub-model 1*. The *Network sub-model 2* offers P_{L_2} and D_2 to *TCP sub-model 3*. Since *TCP sub-model 2* represents connections which cross both bottleneck links, both *Network sub-models 1* and *2* offer the loss probability, $P_{L_1} + P_{L_2} - P_{L_1} P_{L_2}$ to *TCP sub-model 2*. These network sub-models 1 and 2 also give the total delay on both links $D_1 + D_2$ to *TCP sub-model 2*. Since the queueing delays are continuous functions of the queue lengths E_{N_1} and E_{N_2} at the bottleneck links the multi dimensional fixed point procedure between the TCP and network sub-models converges when the values of $P_{L_1}, P_{L_2}, E_{N_1}, E_{N_2}$ and the *ssthresh* probabilities all converge.

8.1.2 A general implementation algorithm of the FPA for multi bottleneck links

As explained in [71] finding the TCP performance metrics using FPA for an arbitrary network topology requires an ingenious strategy. We next present the general implementation algorithm of FPA for a multi bottleneck network with many interacting TCP and network sub-models.

An example of what the input file looks like in this implementation algorithm is given in Figure 8.2. Both the network and TCP sub-models are numbered. For instance as can be seen from the figure, TCP sub-model 1 crosses bottleneck links 1, 2, 4 and 5. TCP sub-models 1, 2 and 3 share the same bottleneck link 2. TCP sub-models 2 and 3 share the bottleneck links 3 and so on.

Algorithm 1 starts by initializing the fixed points and then it iterates until the initialed fixed point variables converge. The function *tcpsm* in the algorithm calculates the load to the respective network sub-model and the function *netism* calculates the packet loss probability and queueing delay to the corresponding TCP sub-model. In the algorithm, *N_TCP_SM*

TCP Sub-model	Routes (with numbered bottleneck links)				
1	1	2	4	5	
2	4	1	3	2	6
3	5	3	2		

Figure 8.2: An example of the input file

and N_{Net_SM} are the numbers of TCP and network sub-models in the network.

8.2 The loss probability on the multi bottleneck links

Figures 8.3 to 8.8 present the loss probability for TCP connections crossing both the MI-NY and POLI-TO bottleneck links. Two dimensional (2-D) plots of the multi bottleneck topology are given in Figures 8.3, 8.4, 8.5 and 8.6. Three dimensional (3-D) plots for the models which give the best prediction of the loss probabilities are shown in Figures 8.7 and 8.8. A buffer capacity of 128 packets, TCP tick values of 500ms and 100ms and the $M/M/1/K$ buffer formula are used. $N1$ refers to the total number of active TCP connections sharing the POLI-TO link, 25% of which are assumed to cross the MI-NY bottleneck link. Here the $0.25N1$ TCP connections are modeled by *TCP sub-model2* and the remaining, $N1 - 0.25N1$ are modeled by *TCP sub-model3*. The connections modeled by *TCP sub-model3* cross only the POLI-TO single bottleneck link. $N2$ represents the total number of active TCP connections sharing the MI-NY link. $N2 - 0.25N1$ of these connections are represented by *TCP sub-model1* and cross only the MI-NY single bottleneck link.

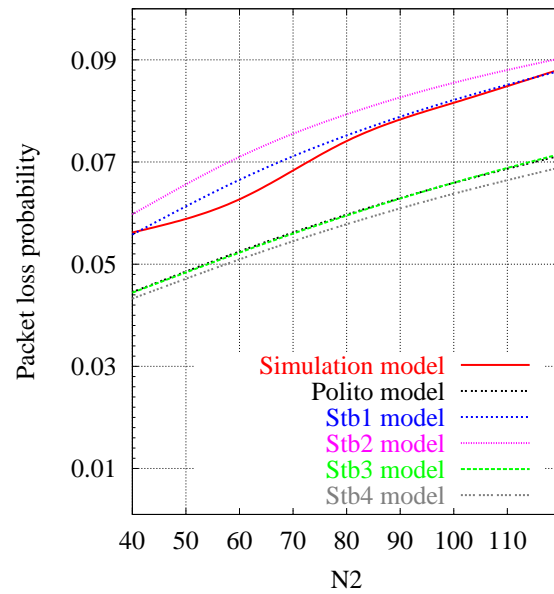
Table 7.1 of loss probabilities for the single network topology holds true for the multi bottleneck case as shown in the figures of this section. When the standard TCP tick value of 500ms is used, our *Stb1* and *Stb2* models outperform both the Polito model and our *Stb3* and *Stb4* models in the estimation of loss probability. When the non-standard TCP tick value of 100ms is used our *Stb3* model slightly outperforms the other models.

Algorithm 1 Implementation of the FPA for a multi bottleneck network

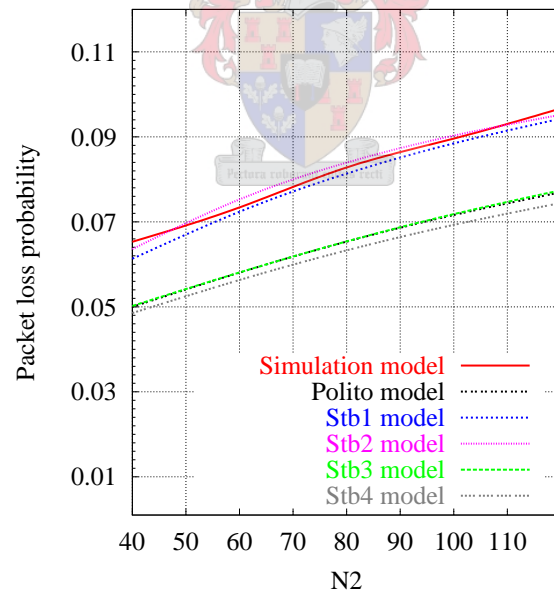
```

// The Initialization
for  $n = 1$  to  $N\_Net\_SM$  do
     $P_{L_n} \leftarrow c_1$  // Some reasonable constant packet loss probability
     $P_{S_n} \leftarrow 1 - P_{L_n}$  // Constant packet success probability
     $E_{N_n} \leftarrow c_2$  // Some reasonable constant queue length
     $D_n \leftarrow E_{N_n}/C_n$  // Queueing delay as a function of the queue length  $E_{N_n}$  and the link
    capacity,  $C_n$ 
     $\Lambda_n \leftarrow 0$  // The total load offered to bottleneck link  $n$ 
end for
for  $t = 1$  to  $N\_TCP\_SM$  do
    read  $h_{t1}$  from the input file // The first bottleneck hop of TCP sub-model  $t$ 
     $l \leftarrow h_{t1}$ 
     $p_{s_t} \leftarrow P_{S_l}$  // Packet success probability of TCP sub-model  $t$  in its path
     $d_t \leftarrow D_l$  // Total delay of TCP sub-model  $t$  in its path
    read  $nh_t$  from input file // Number of bottleneck links (hops) of TCP sub-model  $t$ 
    for  $i = 2$  to  $nh_t$  do
        read  $h_{ti}$  // The number given to the bottleneck link of TCP sub-model  $t$  at hop  $i$ 
         $l \leftarrow h_{ti}$ 
         $p_{s_t} \leftarrow p_{s_t} \times P_{S_l}$ 
         $d_t \leftarrow d_t + D_l$ 
    end for
     $p_{l_t} \leftarrow 1 - p_{s_t}$  // Packet loss probability of TCP sub-model  $t$  in its path
end for
// The main part
repeat
    for  $t = 1$  to  $N\_TCP\_SM$  do
         $l \leftarrow h_{t1}$ 
         $\lambda_{tl} \leftarrow tcpsm(p_{l_t}, d_t)$  // Load offered to link  $l$  by TCP sub-model  $t$ 
         $p_{s_t} \leftarrow P_{S_l}$ 
         $d_t \leftarrow D_l$ 
         $\Lambda_l \leftarrow \Lambda_l + \lambda_{tl}$ 
        for  $i = 2$  to  $nh_t$  do
             $l \leftarrow h_{ti}, l' \leftarrow h_{t(i-1)}$ 
             $\lambda_{tl} \leftarrow \lambda_{tl'} \times P_{S_{l'}}$ 
             $p_{s_t} \leftarrow p_{s_t} \times P_{S_l}$ 
             $d_t \leftarrow d_t + D_l$ 
             $\Lambda_l \leftarrow \Lambda_l + \lambda_{tl}$ 
        end for
         $p_{l_t} \leftarrow 1 - p_{s_t}$ 
    end for
    for  $n = 1$  to  $N\_Net\_SM$  do
         $P_{L_n} \leftarrow netism(\Lambda_n), P_{S_n} \leftarrow 1 - P_{L_n}$ 
         $E_{N_n} \leftarrow netism(\Lambda_n), D_n \leftarrow E_{N_n}/(\Lambda_n \times P_{S_n})$  // Little's Theorem
         $\Lambda_n \leftarrow 0$ 
    end for
until all the initialized  $P_{L_n}$  and  $E_{N_n}$  values converge

```

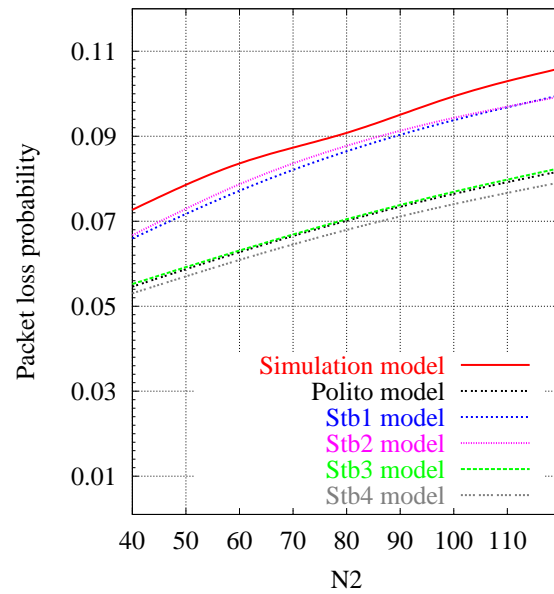


(a) Packet loss probability on the multi bottleneck links: $N_1 = 32$

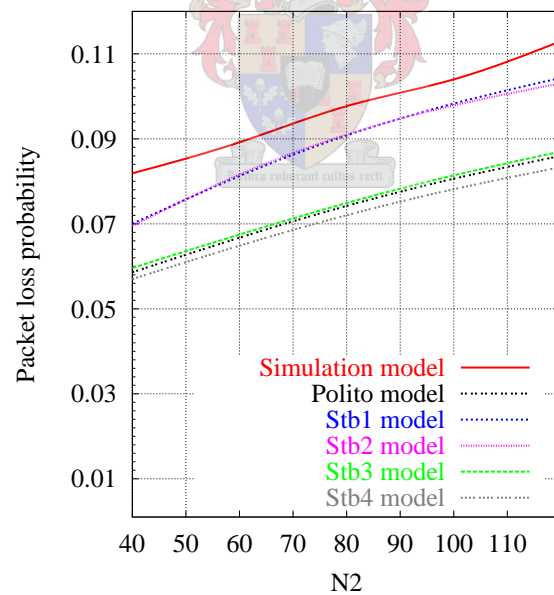


(b) Packet loss probability on the multi bottleneck links: $N_1 = 40$

Figure 8.3: Packet loss probability on the multi bottleneck links when the TCP tick value is 500ms and buffer size is 128 packets

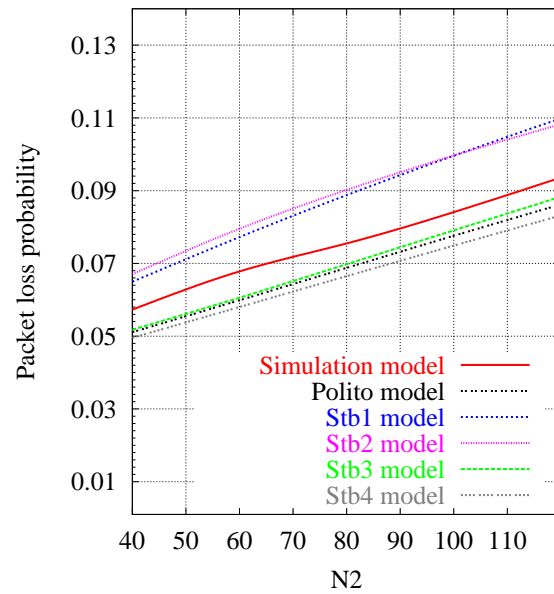


(a) Packet loss probability on the multi bottleneck links: $N_1 = 48$

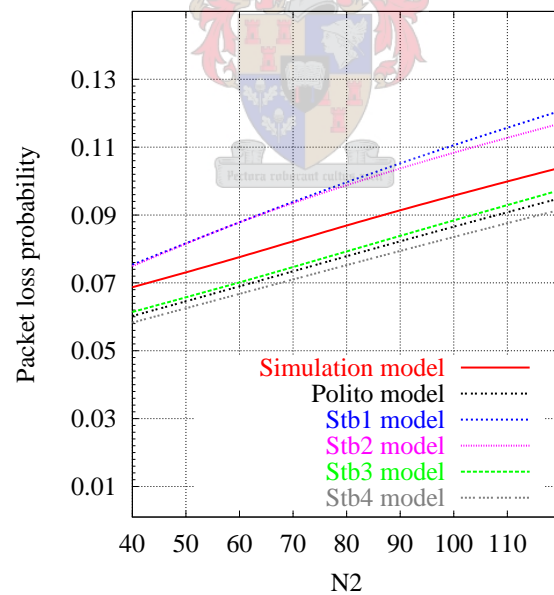


(b) Packet loss probability on the multi bottleneck links: $N_1 = 56$

Figure 8.4: Packet loss probability on the multi bottleneck links when the TCP tick value is 500ms and buffer size is 128 packets

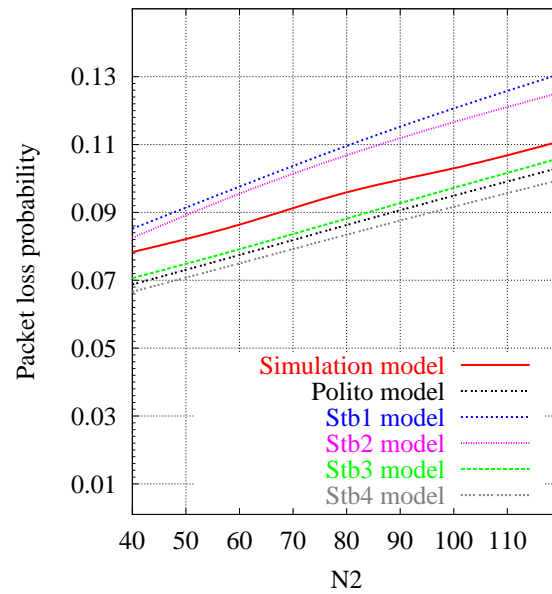


(a) Packet loss probability on the multi bottleneck links: $N1 = 32$

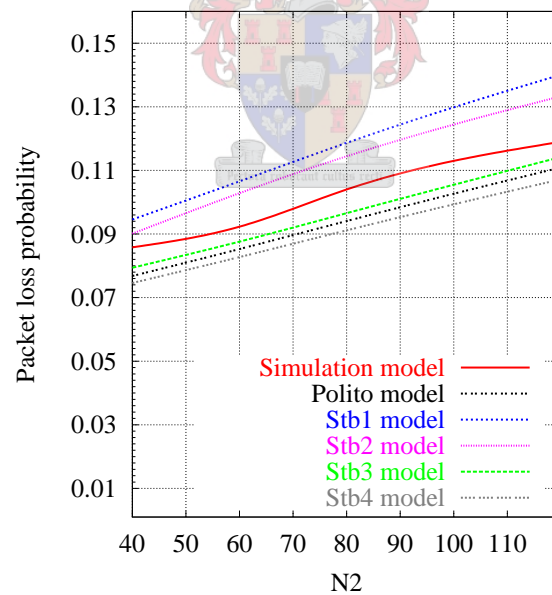


(b) Packet loss probability on the multi bottleneck links: $N1 = 40$

Figure 8.5: Packet loss probability on the multi bottleneck links when the TCP tick value is 100ms and buffer size is 128 packets



(a) Packet loss probability on the multi bottleneck links: $N_1 = 48$



(b) Packet loss probability on the multi bottleneck links: $N_1 = 56$

Figure 8.6: Packet loss probability on the multi bottleneck links when the TCP tick value is 100ms and buffer size is 128 packets

8.3 The average *cwnd* on the multi bottleneck links

Figures 8.11 to 8.14 present the average *cwnd* size values obtained using the *Stb1*, *Stb2*, *Stb3* and *Stb4* models as a function of the number of active TCP connections on both the MI-NY bottleneck link (*N2*) and the POLI-TO bottleneck link (*N1*). In these experiments the $M/M/1/K$ buffer formula, TCP tick values of 500ms and 100ms and buffer capacity of 128 packets are used. We present two dimensional (2-D) plots of the multi bottleneck links on Figures 8.9, 8.10, 8.11 and 8.12. Three dimensional (3-D) plots of the average *cwnd* of some selected models are also given in Figures 8.13 and 8.14. From the figures it can be seen that our models give accurate predictions of the average *cwnd* values.

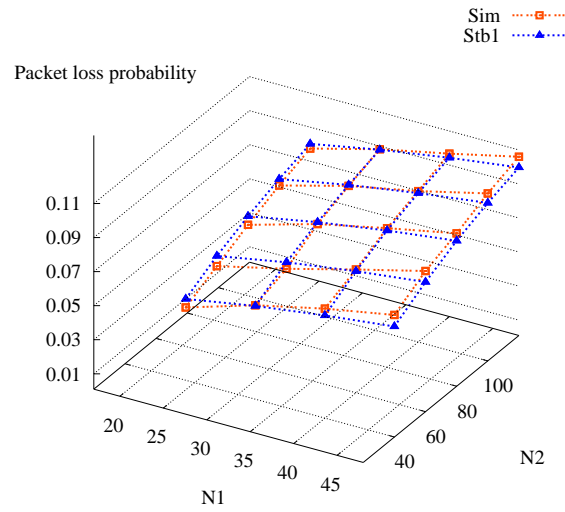
8.4 An analysis of fat pipes

The fact that our analytical models are fast and accurate allows us to obtain results where simulation fails due to very large CPU or memory requirements. Running simulations is difficult when the number of active TCP connections is more than 500. With the growth of link speeds and the number of active TCP connections it becomes impossible to use simulation to get the required performance metrics. For example when the POLI-TO link capacity is 100Mb/sec and the MI-NY link capacity is 1Gb/sec, the simulation model becomes too large to run on a desktop PC. However our analytical models can give results as shown in Figure 8.15. Figures 8.15(a) and 8.15(b) show three-dimensional (3-D) plots of the loss probability and the average *cwnd* size of the fat pipes respectively. Note that a logarithmic scale is used in the plots.

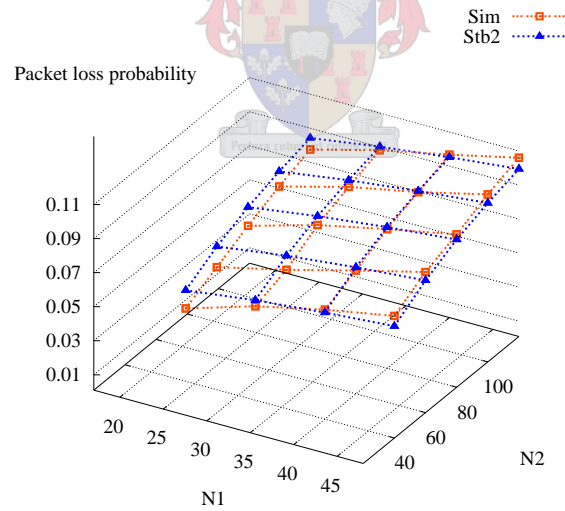
8.5 Chapter summary

This chapter

1. presents an implementation method and algorithm of the FPA (fixed point algorithm) for analytical models of TCP performance for an arbitrary network topology (multi bottleneck network),
2. shows that our *Stb1* and *Stb2* models give even more accurate predictions of the packet loss probability for a multi bottleneck case than our *Stb3* and *Stb4* models and the Polito model when the standard TCP tick value of 500ms is used,

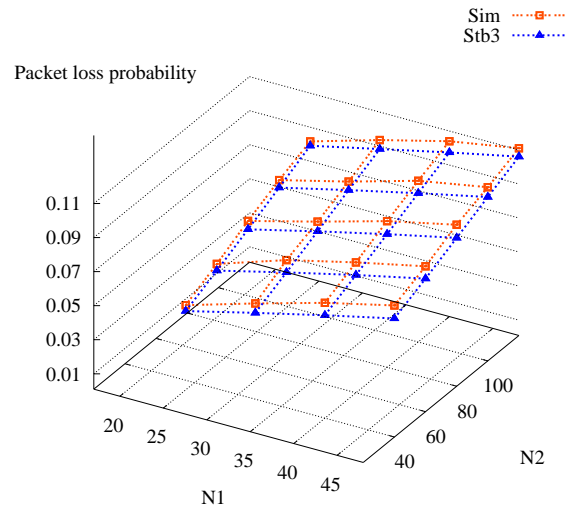


(a) Packet loss probability on the multi bottleneck links using the *Stb1* model

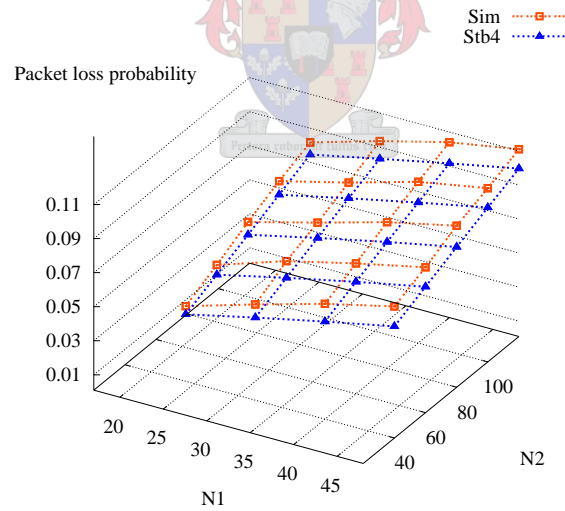


(b) Packet loss probability on the multi bottleneck links using the *Stb2* model

Figure 8.7: Packet loss probability on the multi bottleneck links when a TCP tick value of 500 ms and buffer size of 128 packets are used

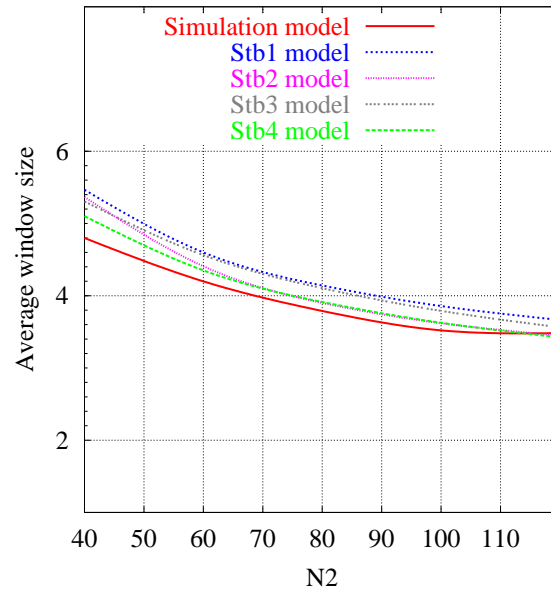


(a) Packet loss probability on the multi bottleneck links using the *Stb3* model

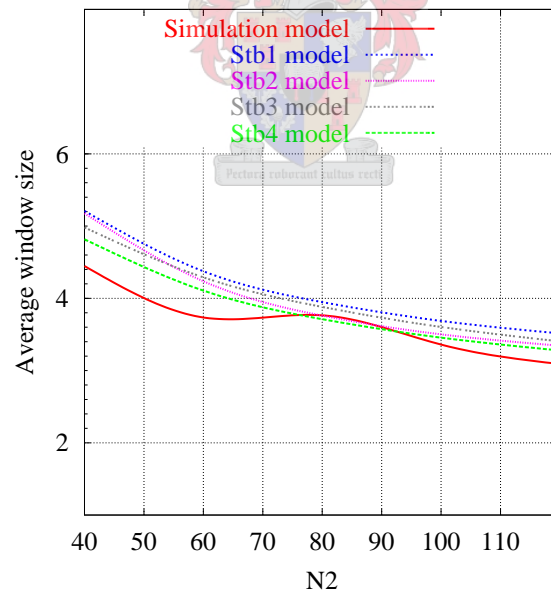


(b) Packet loss probability on the multi bottleneck links using the *Stb4* model

Figure 8.8: Packet loss probability on the multi bottleneck links when a TCP tick value of 100 ms and buffer size of 128 packets are used

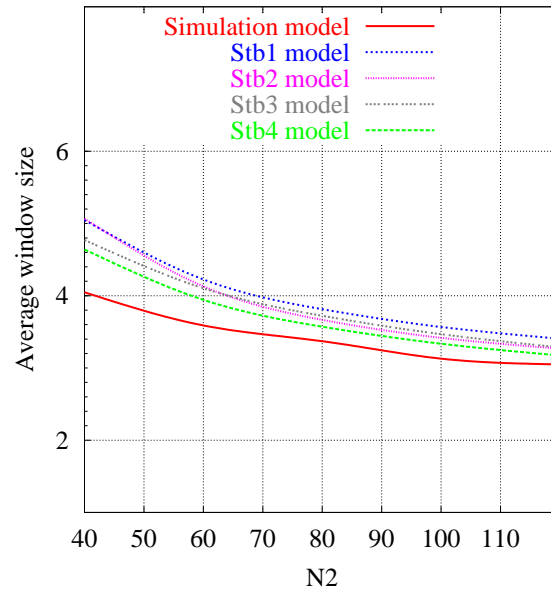


(a) Average *cwnd* on the multi bottleneck links:
 $N1 = 32$

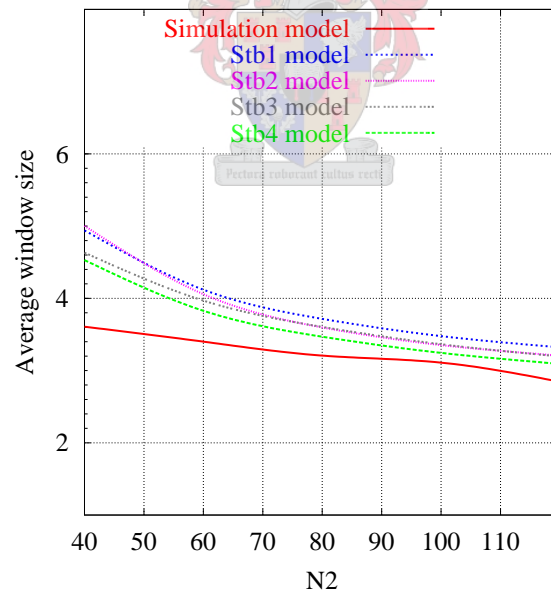


(b) Average *cwnd* on the multi bottleneck links:
 $N1 = 40$

Figure 8.9: Average *cwnd* on the multi bottleneck links when the TCP tick value is 500ms and the buffer size is 128 packets

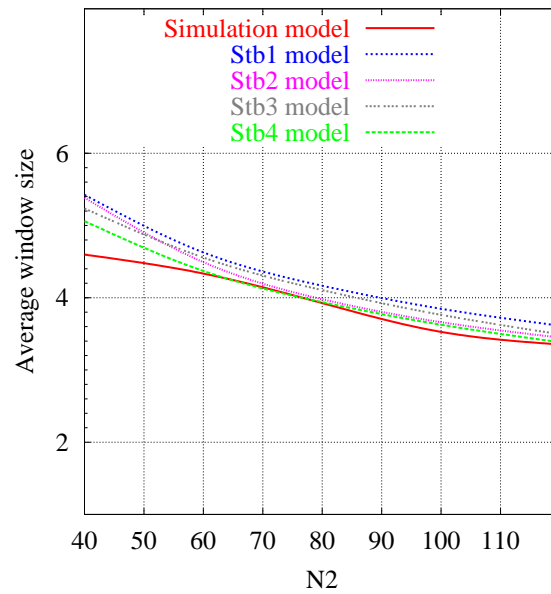


(a) Average *cwnd* on the multi bottleneck links:
 $N1 = 48$

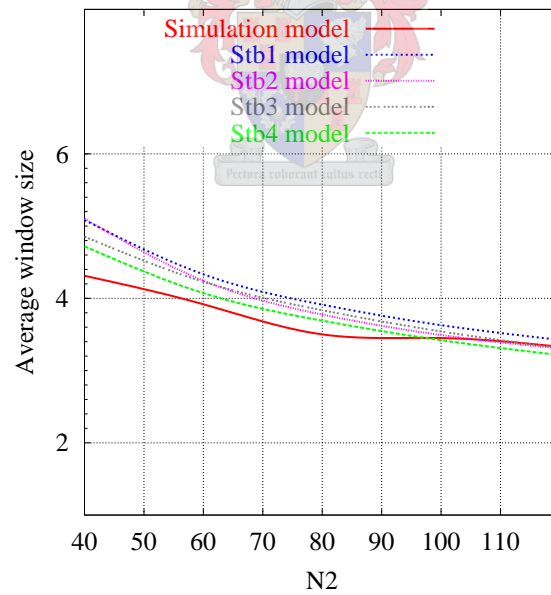


(b) Average *cwnd* on the multi bottleneck links:
 $N1 = 56$

Figure 8.10: Average *cwnd* on the multi bottleneck links when the TCP tick value is 500ms and the buffer size is 128 packets

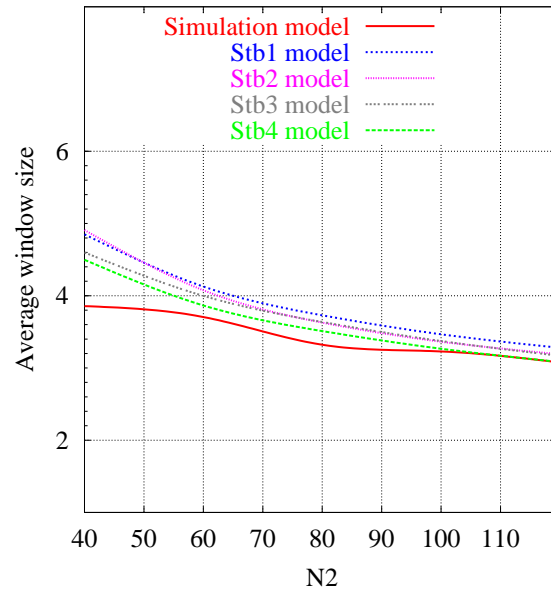


(a) Average *cwnd* on the multi bottleneck links:
 $N1 = 32$

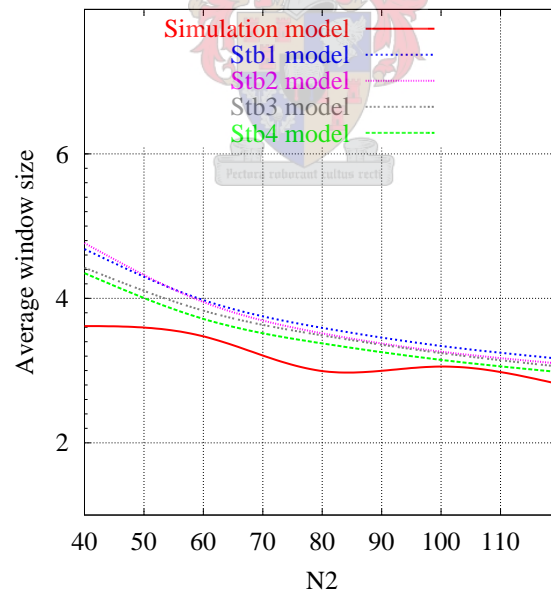


(b) Average *cwnd* on the multi bottleneck links:
 $N1 = 40$

Figure 8.11: Average *cwnd* on the multi bottleneck links when the TCP tick value is 100ms and the buffer size is 128 packets



(a) Average *cwnd* on the multi bottleneck links:
 $N1 = 48$



(b) Average *cwnd* on the multi bottleneck links:
 $N1 = 56$

Figure 8.12: Average *cwnd* on the multi bottleneck links when the TCP tick value is 100ms and the buffer size is 128 packets

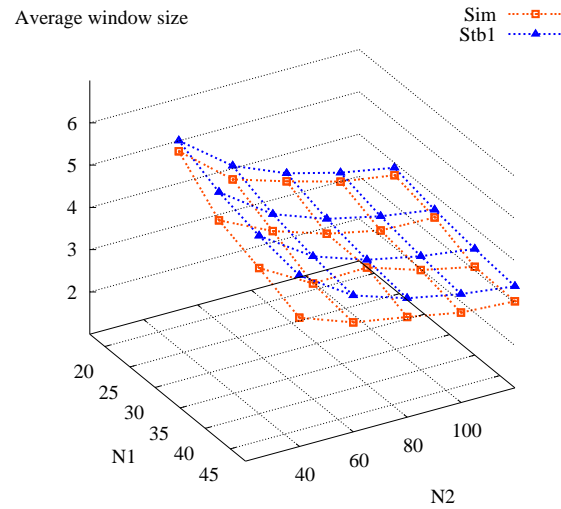
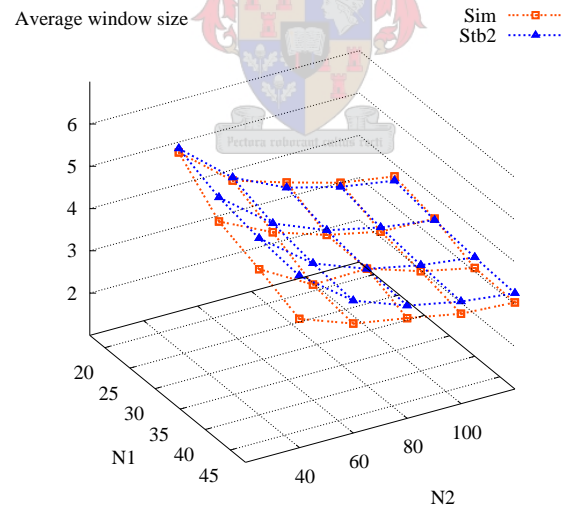
(a) Average *cwnd* using *Stb1*(b) Average *cwnd* using *Stb2*

Figure 8.13: Average *cwnd*: the buffer capacity is 128 packets and the TCP tick value is 500 ms

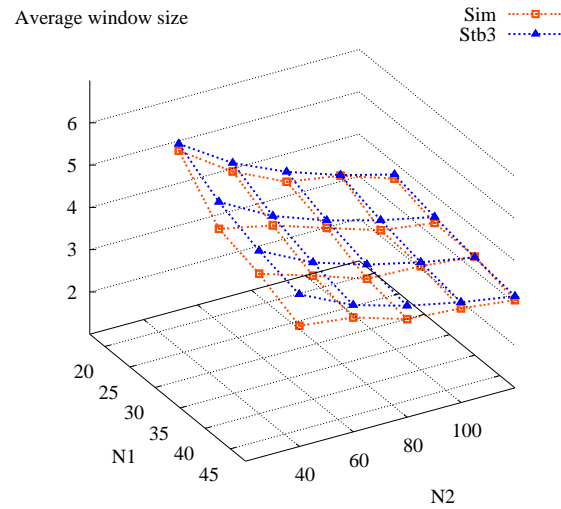
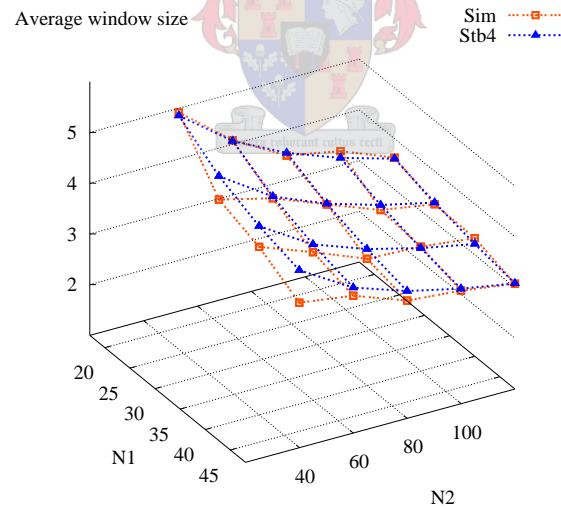
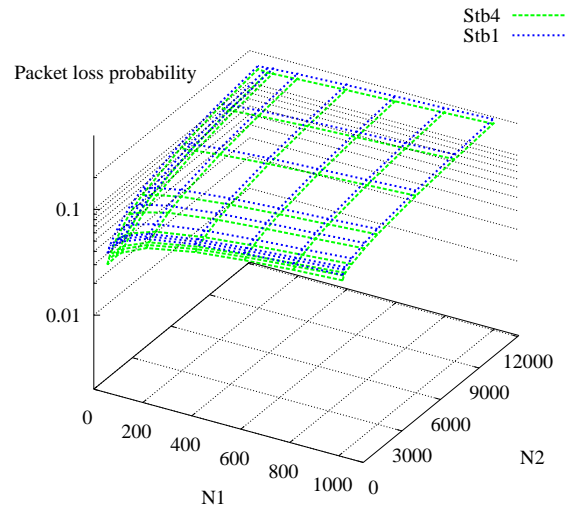
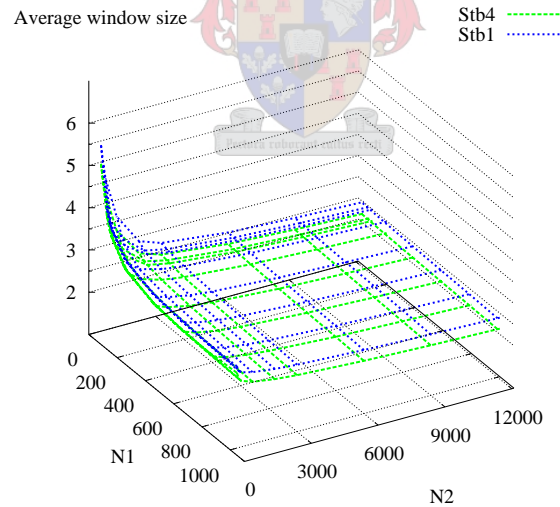
(a) Average *cwnd* using *Stb3*(b) Average *cwnd* using *Stb4*

Figure 8.14: Average *cwnd* on the multi bottleneck links: the buffer capacity of 128 packets and the TCP tick value is 100 ms



(a) multi bottleneck packet loss probability on fat pipes using *Stb1*

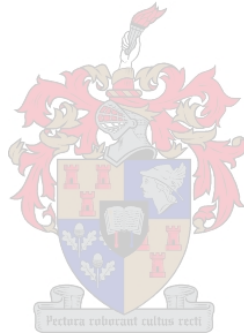


(b) multi bottleneck average window size on fat pipes using *Stb1*

Figure 8.15: Multi bottleneck performance metrics when a TCP tick value of 500 ms and a buffer capacity of 512 packets are used

3. shows that our models give accurate predictions of the average *cwnd* for multi bottleneck networks and
4. shows that our fast models allow us to obtain TCP performance metrics for fat pipes and a more complex networking scenario where simulation fails due to very large CPU and memory requirements.

Our *Stb1* and *Stb4* models along with the numerical results validating them are also (exclusively) presented in [31] and [32] respectively. Another network topology described in Appendix F was also used to further validate our models.



Chapter 9

Thesis summary and work in progress

9.1 Thesis summary

This thesis is concerned with analytical models of TCP performance and the fixed point algorithm (FPA) used to solve them. The analytical models are composed of TCP sub-models which interact with network sub-models in an iterative fixed point procedure.

We present a literature review of the TCP/IP protocol, analytical models of TCP performance and the related mathematical background.

We show how the FPA for analytical models of reliable Internet protocols such as TCP converges to a unique fixed point. Our proof is simple, robust and elegant so that its basic principles apply to both short and long lived connections, to single and multi bottleneck networks and to AQM and Drop Tail routers in different networking scenarios. The concepts used in the proof specify the necessary conditions in order to use the FPA to solve analytical models of reliable Internet protocols such as TCP.

We investigate the Polito model. Based on the protocol description we give detailed analysis, explanation and derivation of all formulas used in the Polito model in order to apply them to the more efficient models we develop (see the appendices of this thesis).

We develop several 6 state analytical models of TCP performance namely *Stb1*, *Stb2*, *Stb3* and *Stb4*. Our *Stb1* and *Stb2* models give more accurate predictions of the packet loss probability (and hence capture the TCP burstiness better) than the Polito model ([39]) and our other models when the standard TCP tick value of 500ms is used. Our *Stb4* model gives

more accurate predictions of the *cwnd* size distribution than our other models. These different performance achievements of our models arise from the different natures of the generalized bounded and truncated geometric distributions on which our *Stb1* and *Stb4* models are based respectively. Our *Stb2* and *Stb3* models are based on and reflect the behaviors of both distributions. We have also shown that all our models give accurate predictions of average *cwnd*, *ssthresh* and throughput values.

We show how the geometric, generalized bounded and truncated geometric distributions can be used to model TCP/IP networks. We model the packet loss correlation and the TCP sliding window algorithm.

We give closed form expressions for many important TCP performance values and distributions. These include the conditional *SS*, *CA*, the *cwnd*, the *ssthresh* and the retransmission probability distributions, the TO and TD probability values. We also derive a closed form expression for the probability that TCP is in the *SS* transmission and retransmission states when the *cwnd* is 1 from which the other probabilities that TCP is in a given transmission or retransmission state with a corresponding *cwnd* value can be found.

We show how the $M/M/1/K$ and the geometric techniques of the network sub-models are combined to obtain a numerically more efficient method of solving the network sub-models.

The complexity of our models is less than that of the Polito model. Hence our models can be used to rapidly analyze network topologies with many bottleneck links. Large capacity paths and more complex network scenarios where simulation fails due to very high CPU and memory requirements can easily be analyzed using our models. The low computational complexity of our models can also be exploited to solve the limitation of the Polito model in computing the throughput and other metrics of individual TCP connections as shown in section 6.4.

We develop an implementation method and algorithm of the FPA for analytical models of TCP performance for an arbitrary network topology (multi bottleneck network).

Our models show that the timeout probability is higher when a smaller buffer capacity of 128 packets and a higher TCP tick value of 500ms are used. When the distance of TCP connections is long a smaller router buffer capacity of 128 packets and the standard TCP tick value of 500ms result in a higher throughput. When the distance of the connections is short a smaller TCP tick value of 100ms with the buffer capacity of 128 packets gives a better throughput. It seems that a larger tick value is better for long TCP connections and a smaller tick for short TCP connections. However when a larger buffer capacity of 512 packets

is used, both tick values give similar results. Our results also show that the variation of the throughput values of individual TCP connections sharing the same bottleneck links is higher when the router buffer capacity is smaller (128 packets).

9.2 Work in progress

We are combining our convergence and uniqueness proof of the FPA of analytical models of TCP with the network optimization and equilibrium studies of Low *et al.* [75, 85, 86, 87, 93] to analyze the equilibrium, stability and global uniqueness issues of TCP, other reliable protocols and the Internet as a whole. We will investigate more findings ([43]) on topology and functional analysis about the FPA. For instance a contraction mapping has a unique fixed point.

We are extending the modeling techniques used in this thesis to other TCP implementations, to short-lived TCP connections, to other reliable Internet protocols and non-responsive UDP (user datagram protocol) flows in wireless and other networking scenarios.

In appendix E we derive expressions for the *cwnd* size probability distribution conditioned on the *ssthresh* probability distribution and vice-versa based on the generalized bounded geometric distribution discussed in chapter 6. We will investigate whether a similar technique based on the generalized truncated geometric distribution gives more accurate predictions of the performance metrics. We will use the *cwnd* and *ssthresh* distributions with the Markov process model [42] and Markov chain model [56] for hybrid space-terrestrial networks where the states are represented by a set of window size and *ssthresh* values in order to get more efficient models of the Internet.

Our modeling framework can also be used with the simpler transition rates presented in [23] which are similar to those given in [101] with small modifications.

In this thesis we have used an $M/M/1/K$ queueing system for the network sub-models. The low computational complexity of our models allows us to use more complex network models with batch arrivals and services to obtain a more accurate estimation of the packet loss probabilities and thereby better model the burstiness of TCP.

Appendix A

The derivation of P_{L_f}

The derivation of the probability P_{L_f} that the first packet in a sliding window (see Figure B.4) is lost is based on the fact that the loss rate ΛP_L obtained from the network sub-model in the iterative fixed point procedure is identical with the average loss rate using the loss correlation model described in section 6.1.2. As shown in Equation D.4 we have

$$\Lambda = \sum_{i=1}^{W_m} i \lambda^*(i)$$

where $\lambda^*(i)$ is the rate at which batches of i packets are sent into the network in each RTT . An expression for $\lambda^*(i)$ is given in Equation D.5.

Suppose the first packet loss occurs at position j in a batch of packets of size i . With reference to Equation A.2 the summation over j considers all positions of the first loss within a batch of size i . The first $j - 1$ packets are subject to no loss. The probability that the j th packet is lost is $P_{S_f}^{j-1} P_{L_f}$. The probability of any one of the remaining $i - j$ packets being lost is $P_{L_a} = \alpha P_{L_f}$.

Given that the first loss occurred at position j , the conditional expected number of lost packets is

$$1 + \sum_{r=1}^{i-j} r \binom{i-j}{r} P_{L_a}^r P_{S_a}^{i-j-r} = 1 + (i-j) P_{L_a}. \quad (\text{A.1})$$

Thus

$$\Lambda P_L = \sum_{i=1}^{W_m} \lambda^*(i) \sum_{j=1}^i P_{S_f}^{j-1} P_{L_f} (1 + (i-j) \alpha P_{L_f}). \quad (\text{A.2})$$

Re-arranging the terms of Equation A.2

$$\Lambda P_L = \sum_{i=1}^{W_m} \lambda^*(i) \left(\sum_{j=1}^i P_{S_f}^{j-1} P_{L_f} + \alpha P_{L_f} i \sum_{j=1}^i P_{S_f}^{j-1} P_{L_f} - \alpha P_{L_f}^2 \sum_{j=1}^i j P_{S_f}^{j-1} \right). \quad (\text{A.3})$$

But

$$\sum_{j=1}^i P_{S_f}^{j-1} P_{L_f} = 1 - P_{S_f}^i \quad (\text{A.4})$$

and

$$\sum_{j=1}^i j P_{S_f}^{j-1} = \frac{P_{S_f}^i (i P_{S_f} - i - 1) + 1}{(P_{S_f} - 1)^2}. \quad (\text{A.5})$$

Using Equations A.4 and A.5, Equation A.3 becomes

$$\begin{aligned} \Lambda P_L &= \sum_{i=1}^{W_m} \lambda^*(i) \left(1 - P_{S_f}^i + \alpha P_{L_f} i (1 - P_{S_f}^i) - \alpha P_{L_f}^2 \frac{P_{S_f}^i (i P_{S_f} - i - 1) + 1}{(P_{S_f} - 1)^2} \right) \\ &= \sum_{i=1}^{W_m} \lambda^*(i) \left(1 - P_{S_f}^i + \alpha P_{L_f} i - \alpha P_{L_f} i P_{S_f}^i - \alpha P_{S_f}^i (i P_{S_f} - i - 1) - \alpha \right) \\ &= \alpha P_{L_f} \sum_{i=1}^{W_m} i \lambda^*(i) + \sum_{i=1}^{W_m} \lambda^*(i) \left(1 - P_{S_f}^i - \alpha i (1 - P_{S_f}) P_{S_f}^i - \alpha P_{S_f}^i (i P_{S_f} - i - 1) - \alpha \right) \\ &= \Lambda \alpha P_{L_f} + (1 - \alpha) \sum_{i=1}^{W_m} \lambda^*(i) (1 - P_{S_f}^i) \\ &= \Lambda \alpha P_{L_f} + (1 - \alpha) \sum_{i=1}^{W_m} \lambda^*(i) (1 - (1 - P_{L_f})^i). \end{aligned}$$

Define

$$f(P_{L_f}) = (\alpha P_{L_f} - P_L) \Lambda + (1 - \alpha) \sum_{i=1}^{W_m} \lambda^*(i) (1 - (1 - P_{L_f})^i) = 0. \quad (\text{A.6})$$

P_{L_f} is obtained using the Newton-Raphson method (see section 3.9 of [28] and [99]) by the iterative equation

$$P_{L_f}^{(n+1)} = P_{L_f}^{(n)} - \frac{f(P_{L_f}^{(n)})}{f'(P_{L_f}^{(n)})} \quad (\text{A.7})$$

where $P_{L_f}^{(n)}$ and $P_{L_f}^{(n+1)}$ are the P_{L_f} values at iterations n and $n+1$ respectively. The iteration is continued until $|P_{L_f}^{(n+1)} - P_{L_f}^{(n)}| < 10^{-8}$. The Newton-Raphson procedure is initialized by setting $P_{L_f}^{(0)}$ to P_L .

Appendix B

State transitions

This chapter presents explanations and derivations of the formulas used in the preceding chapters. It is organized as follows. In sections B.1 and B.2 we present the detailed state transition graphs and the idealized TCP transmission cycle. Section B.3 gives derivations and explanations of the detailed transition probabilities used in our study. The chapter summary is given in section B.4.

B.1 The detailed state transition graph

Let \mathcal{T} denote the set of TCP states given in section 6.1.2. A transition from a state $A \in \mathcal{T}$ into a state $B \in \mathcal{T}$ is any of the transitions from A_i (state A with a *cwnd* value i) to B_j (state B with a *cwnd* value j). For instance a transition from SS into CA is any of the transitions from SS_i when the *cwnd* value is i , into CA_i when the *cwnd* is i . A transition from a state A to itself, which can occur in SS and CA is any of the transitions from A_i to A_{i+1} . To calculate these transition probabilities from one state A into the same or another state, a weighted average of each transition when the states assume the same or different *cwnd* values is taken. The derivation of the weights is given in Equation 6.14.

The detailed state transition diagram of the 6 states with their specific congestion window sizes is shown in Figure B.1. The figure shows the *steady-state* behavior of a TCP connection when the maximum window size $W_m = 10$ segments. The thin lines describe transitions from one state with a certain *cwnd* value to the same or another state with a certain *cwnd* value. The thick lines represent transitions from a state with every possible *cwnd* value to a state with a *cwnd* value of 1.

Section B.2 explains how the *cwnd* grows in the *SS* and *CA* states. Why and how TCP transitions from one state with a certain *cwnd* value to the same or another state with the same or another *cwnd* value is given in section B.3.3. The section for example explains why a TCP connection which was in the *SS*₃ state transitions into the *SST*₃, *SST*₄ or the *SST*₅ state.

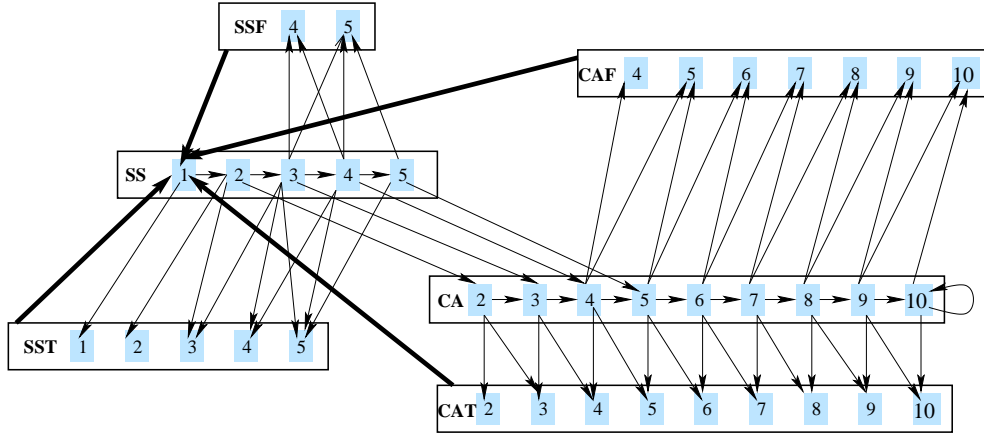


Figure B.1: A detailed description of the transitions of the TCP model where the exponential back-off is built into the *SS* and *SST* states of our models

The retransmission, exponential back-off and Karn's algorithm states of [39] are built into the *SS* and *SST* queues described in Figure B.1. If TCP is in the *SST*, or *CAT* states it goes to *SS*₁ to retransmit the lost packet. If this first retransmission is successful TCP increases the *cwnd* by one and goes to *SS*₂. Otherwise it goes to *SST*₁ where it waits for the first retransmission timeout to expire. This retransmission timeout is obtained by doubling the transmission timeout. After the first retransmission timeout expires TCP goes back to *SS*₁ and retransmits the lost packet for the second time by doubling the timeout value of the first retransmission. If this retransmission is not successful, TCP goes to the next retransmission until it finally reaches the maximum number of consecutive retransmissions *c* allowed before closing the connection. If the retransmission is successful, TCP goes to *SS*₂ state. It then transmits two packets. If the first packet is lost TCP goes to *SST*₂ and if the second packet is lost it goes into *SST*₃ to wait for the corresponding timeout values to expire. These timeout values are calculated based on the doubled timeout values of the first retransmission under the influence of the Karn's algorithm as shown in appendix C. After these timeout expire TCP goes back into *SS*₁ of the same retransmission phase.

In Figure B.2 the SS_i^j and SST_i^j queues represent the *SS* and *SST* queues when the *cwnd* size is *i* at retransmission *j*. From the above explanation it can be seen that the SST_i , $1 \leq i \leq 3$

and the SS_i , $1 \leq i \leq 2$ queues of Figure B.1 are aggregates of the SST_i^j , $1 \leq i \leq 3$, $0 \leq j \leq c$ and SS_i^j , $1 \leq i \leq 2$, $0 \leq j \leq c$ queues of Figure B.2 respectively. The service times and the loads offered by the SST_i queues are calculated as the weighted averages of the corresponding SST_i^j values as shown in sections 6.1.5 and 6.1.7 respectively. The weights are the probabilities that TCP is in the respective retransmission states as shown in section 6.1.4.

Figure B.2 shows the transmission and retransmission states separately. In the figure subscript (i) and superscript (j) are used to identify the *cwnd* and retransmission indices respectively. If timeout loss occurs in *SS* or *CA* TCP goes to the SS_1^1 state. It retransmits the lost packet. If the retransmitted packet is successful, then TCP goes to the SS_2^1 state. If the first retransmission is not successful, TCP goes to SST_1^1 where it waits for the timeout value of the first retransmission to expire. After this timeout expires TCP goes to the SS_1^2 state to retransmit the lost packet for the second time. If this retransmission is successful, TCP goes to SS_2^2 where it sends two new packets. If the first of these two new packets is not successful TCP goes to the SST_2^2 state. If the second packet fails it goes into the SST_3^2 state where it waits for the timeout to expire. After the expiration of this timeout value TCP goes back to SS_1^2 . More explanation about how and why TCP transitions from one state into another state is given in section B.3.3.

B.2 The idealized TCP transmission cycle

Figure B.3 which is used in [38] shows an idealized TCP transmission cycle when $ssthresh = 6$. TCP begins in slow start, and continues into congestion avoidance. Time evolves along the horizontal axis from left to right, wrapping around when the line ends. Each line represents one *RTT*. Larger rectangles above the lines represent transmitted TCP segments with their sequence number¹. The number above the segments is the congestion window size at the time of transmission of that segment. The small rectangles below the time line represent ACKs received by the transmitter.

During the first *RTT* of *SS* a TCP connection offers 1 packet to the network, and when the first ACK arrives, 2 packets are transmitted back-to-back. ACK 1 and packets 2 and 3 are duplicated on line 1 and 2 to show how the time line wraps around to the next time line of the chart. At this point in the protocol evolution, models such as [63] assume that transmission times are negligible, and ACKs are received back-to-back, so that *cwnd* doubles and 4 TCP segments are sent back-to-back. This assumption is an approximation because

¹For the sake of simplicity we number segments, and not bytes, starting from 1. We number ACKs after the packet they acknowledge, so that segment 1 is acknowledged by ACK 1 as in [38].

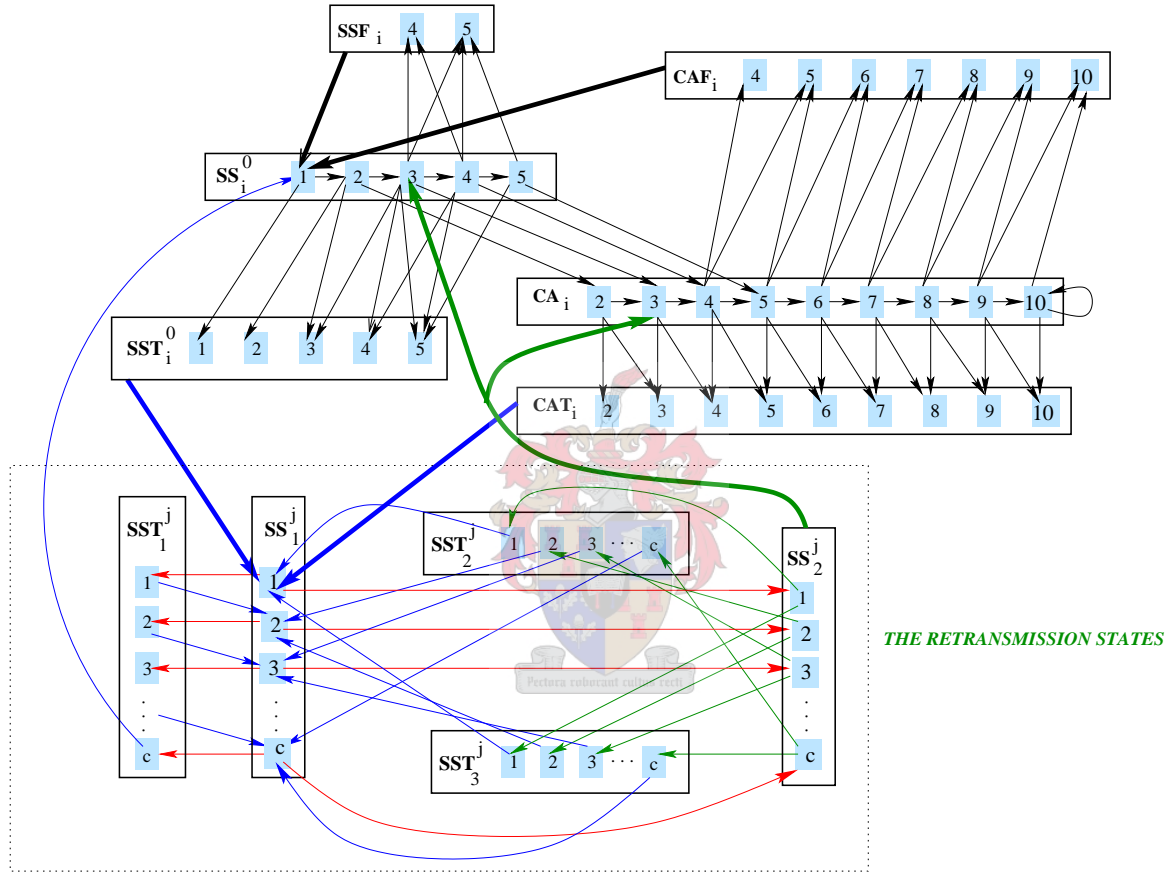
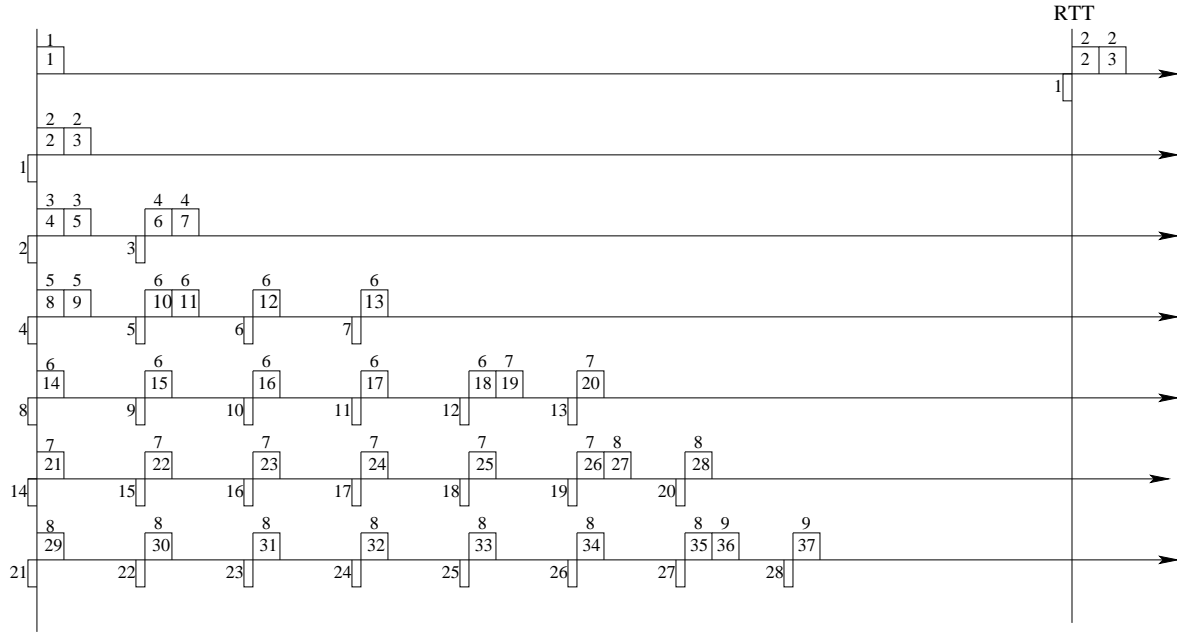


Figure B.2: A description of the transitions of the TCP model when the exponential back-off is separately shown.

Figure B.3: The idealized TCP cycle when $ssthresh = 6$

(1) transmission times are not negligible, (2) the TCP congestion window increases in a continuous manner: hence before growing to 4, the window must assume the value 3 for at least 2 transmission times, and so forth for the other values, and (3) if the network is heavily loaded, then packets and ACKs may be interleaved in the network, and ACKs cannot arrive back-to-back at the source.

When $ssthresh = 6$ is reached as shown in line 4, TCP sends packet 12 in *CA* after the receipt of ACK 6. Then TCP needs to accept 6 ACKs (ACKs 7, 8, 9, 10, 11 and 12) before the *cwnd* grows to 7. At this stage TCP need not wait for another ACK to increase the *cwnd* to 7 as the *cwnd* has to increase by one after the receipt of the ACK of the first packet sent when the *cwnd* was 6. For more information on how the window grows, see chapter 20.7 and related chapters of [83] and Figure B.4.

B.3 The $P(A_i, B_j)$'s and their derivation

The derivation of the probabilities $P(A_i, B_j)$ that TCP progresses from state A_i to B_j requires the timeout and fast retransmit loss and $ssthresh$ formulas described in the following sections.

B.3.1 Timeout and fast retransmit losses

As in [47], for transitions that are consequences of losses, the probability $P_{\text{TO}}(i)$ that a loss is detected by timeout given that i packets were transmitted after the lost packet is the probability that two or fewer ACKs are received for the i packets sent after the first loss. If 3 or more ACKs are received then the loss will be detected by triple duplicate ACKs. If less than 3 packets are sent after the lost packet then the loss must be a timeout (TO) loss as there are not enough packets for a triple duplicate ACK (TD) loss to occur. Therefore

$$P_{\text{TO}}(i) = \sum_{j=0}^2 \binom{i}{j} P_{L_a}^{i-j} P_{S_a}^j \quad (\text{B.1})$$

for $3 \leq i \leq W$ and the corresponding probability $P_{\text{TD}}(i)$ that the loss is detected by triple duplicate ACKs given that i segments were transmitted after the lost segment is

$$P_{\text{TD}}(i) = 1 - P_{\text{TO}}(i).$$

In Equation B.1, the probabilities P_{L_a} and P_{S_a} which are described in section 6.1.2 are used for the losses and successes of packets after the first packet loss respectively. The probabilities, P_{L_f} and P_{S_f} are used for the first packet loss and success probabilities respectively. The other terms used in the calculation of the transition probabilities $P(A_i, B_j)$ are the *ssthresh* formulas given below.

B.3.2 The slow start threshold

When a loss is detected (see section 2.5.3) *ssthresh* is updated according to

$$ssthresh = \max(2, \lfloor \min(W_m, cwnd)/2 \rfloor). \quad (\text{B.2})$$

The loss can be in the *SSF* or *CAF* states which model the triple duplicate ACK losses or in the *SST* or *CAT* states which model timeout losses in the states *SS* and *CA* respectively. The distribution of *ssthresh* the *SS* threshold that discriminates between the *SS* and *CA* window growth and which is used in the calculation of the transition probabilities is obtained by the formulas given in Equation B.3 below.

Let $\mathcal{F}_i = \{SST_i, SSF_i, CAT_i, CAF_i\}$ denote the set of timeout and triple duplicate acknowledgment states when the congestion window size is i . The total arrival rate, $\lambda(\mathcal{F}_i)$ of packets into the states \mathcal{F}_i is given by

$$\lambda(\mathcal{F}_i) = \sum_{A \in \mathcal{F}_i} \lambda(A)$$

where the arrival rate $\lambda(A)$ to state (queue) A is given by Equation 6.17.

The $P_t(i) = P(\text{ssthresh} = i)$ is

$$P_t(i) = \begin{cases} \frac{\sum_{j=1}^5 \lambda(\mathcal{F}_j)}{\sum_{j=1}^{W_m/2} \lambda(\mathcal{F}_j)}, & i = 2 \\ \frac{\lambda(\mathcal{F}_{2i}) + \lambda(\mathcal{F}_{2i+1})}{\sum_{j=1}^{W_m/2} \lambda(\mathcal{F}_j)}, & 3 \leq i \leq W_m/2. \end{cases} \quad (\text{B.3})$$

$P_t(i)$ is initially set to $2/W_m$ at the beginning of the iterative fixed point procedure and calculated by Equation B.3 for the other iterations. Equation B.3 is a consequence of Equation B.2 which implies that $cwnd$ must grow to $2i$ or $2i + 1$ in order for the ssthresh to be i , and $\text{ssthresh} \geq 2$.

The other ssthresh probabilities used in the calculation of the transition probabilities are

- $P_t(i, i) = h(i) = P(\text{ssthresh} = i \mid \text{ssthresh} \geq i)$ which is given by

$$P_t(i, i) = \frac{P_t(i)}{1 - F_t(i)}. \quad (\text{B.4})$$

where $F_t(i)$ is the ssthresh cumulative distribution. $h(i)$ is the *failure* or *hazard* rate function (see section 4.8) where the survival function $S(i) = 1 - F(x)$. Failure in this case happens when ssthresh is reached.

- $P_t(j, i) = P(\text{ssthresh} = j \mid \text{ssthresh} \geq i)$ is given by

$$P_t(j, i) = \frac{P_t(j)}{1 - F_t(i)}. \quad (\text{B.5})$$

- $P_c(j, i) = P(\text{ssthresh} \geq j \mid \text{ssthresh} \geq i), j \geq i$ is given by

$$P_c(j, i) = \frac{1 - F_t(j)}{1 - F_t(i)}. \quad (\text{B.6})$$

- $P_c(i + 1, i) = P(\text{ssthresh} \geq i + 1 \mid \text{ssthresh} \geq i)$ is the probability that ssthresh is not reached when $cwnd = i$ and is given by

$$P_c(i + 1, i) = \frac{1 - F_t(i + 1)}{1 - F_t(i)}. \quad (\text{B.7})$$

B.3.3 The transition probabilities and their derivation

The transition probabilities $P(A_i, B_j)$ from state A_i with $cwnd$ size i into state B_j with $cwnd$ size j and their derivations are given below. These transition probabilities are used in [38], [39] and [40] where no derivation is given. Instead of these values, the simpler transition rates presented in [23] which are similar to those given in [101] can also be used in our models with small modifications.

The state transition probabilities: no losses

Figure B.4 describes the TCP sliding window protocol from which the $P(A_i, B_j)$ are derived. The *ssthresh* used in the figure is 4. The ACKs of the packets and the *cwnd* when the packets are sent are given in the left and right columns respectively. As in Figure B.3, the ACKs are numbered after the packet they acknowledge, so that segment 1 is acknowledged by ACK 1. Figure B.4 shows that *cwnd* increases by 1 and the sliding window moves one packet forward sending a total of 2 packets for every ACK received in *SS*. When ACK 3 is received *cwnd* increases by 1, the window slides one packet forward, the *ssthresh* is reached and TCP enters the *CA* state. TCP starts sending a packet in *CA* when ACK 4 is received. TCP sends one packet for each of ACKs 5, 6, 7 and 8 before increasing the *cwnd* by 1 packet. After ACK 8 is received, the sliding window moves one packet forward, packet 12 is sent, *cwnd* increases by 1 packet and packet 13 is sent. The notation 4,5 is used to show that the *cwnd* is 4 packets when packet 12 is sent and 5 packets when packet 13 is sent.

As shown in Figures B.3 and B.4, a TCP connection which is in the *SS* state when the *cwnd* = 1 transmits 1 packet. The probability, $P(SS_1, SS_2)$, that TCP goes from *SS* when the *cwnd* = 1 to *SS* when the *cwnd* = 2 is the probability P_{S_f} that the packet transmitted while in *SS*₁ is successfully received.

For other *cwnd* values in *SS* which represent the transmission of 2 packets following an ACK reception, a TCP connection moves from *SS*_{*i*} when the *cwnd* = *i* to *SS*_{*i*+1} when the *cwnd* = *i* + 1 if both transmissions are successful and the *ssthresh* is not reached (*ssthresh* > *cwnd*). The probability of this event is $P(\text{two packets delivered}) \times P(\text{ssthresh not reached})$

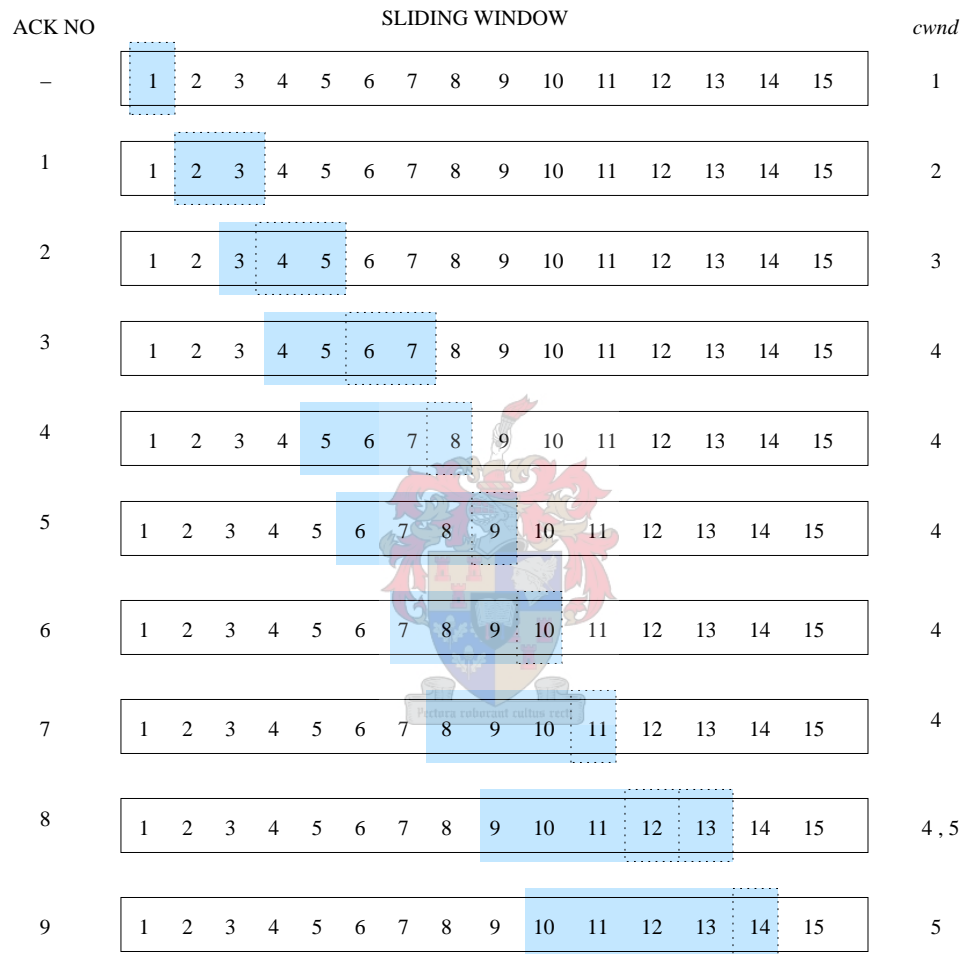
$$P(SS_i, SS_{i+1}) = P_{S_f}^2 P_c(i+1, i), \quad 2 \leq i < W_m/2.$$

As shown in Figures B.3 and B.4, once the *ssthresh* is reached (*ssthresh* = *i*) TCP moves into the *CA*_{*i*} state with probability $P(\text{two packets delivered}) \times P(\text{ssthresh reached})$

$$P(SS_i, CA_i) = P_{S_f}^2 P_t(i, i), \quad 2 \leq i \leq W_m/2.$$

$P_c(i+1, i)$ and $P_t(i, i)$ are derived in Equations B.4 and B.6 respectively and P_{S_f} is described in section 6.1.2.

If *ssthresh* is reached (see lines 4 of Figure B.3), TCP enters *CA* without increasing *cwnd*. Here the ACK of the first TCP segment transmitted in *CA* with size *w* makes the window grow to *w* + 1 after one *RTT*. The number of packets transmitted during *CA* when the *cwnd* = *w* if no loss occurs is *w* + 1 as *w* ACKs have to be received to increase the window by 1. The probability that TCP increases the window size by 1 in *CA*, $P(CA_i, CA_{i+1})$ is

Figure B.4: The sliding window protocol when $ssthresh = 4$

therefore the probability that $i + 1$ packets are successfully transmitted ($P_{S_f}^{i+1}$, $2 \leq i < W_m$). The transition from CA_{W_m} into CA_{W_m} is the probability that W_m packets are successfully transmitted ($P_{S_f}^{W_m}$).

The state transition probabilities: losses in SS

If the ACK of the packet sent in SS_i is not received within one RTT TCP will either enter the SST_j state and wait for the timeout to expire, or it will enter the SSF_j state and wait for the arrival of three duplicate ACKs. The $cwnd$ size j in the SST and SSF states which shows how much the sliding window is opened before the transmitter detects the loss either by fast retransmit or by timeout expiry is influenced by the loss pattern and the $ssthresh$ distribution. Hence the transitions into the SST_j and SSF_j states involve the $ssthresh$ and the loss pattern formulas given in sections B.3.2 and B.3.1 respectively. Transitions when the first packet transmitted with window size i is lost involve the term P_{L_f} and transitions when only the second packet transmitted with window size i is lost involve the expression $P_{S_f} P_{L_f}$.

Losses in SS when $cwnd = i < 3$

Consider the first time line illustrated in Figure B.3 where $cwnd = 1$. If the first packet (packet 1) is lost and ACK 1 is not received within one RTT then TCP will go from state SS_1 to state SST_1 with probability

$$P(SS_1, SST_1) = P_{L_f}. \quad (B.8)$$

Consider the second time line illustrated in Figure B.3 where $cwnd = 2$. The following loss events are possible

- If the first packet (packet 2) is lost and ACK 2 is not received within one RTT then $cwnd$ is not increased. TCP will go from state SS_2 to state SST_2 with probability

$$P(SS_2, SST_2) = P_{L_f}. \quad (B.9)$$

- If packet 2 is successful and packet 3 is lost so that ACK 3 is not received within one RTT then

- if $ssthresh = 2$ then $cwnd$ is not increased and TCP will go from state SS_2 to state SST_2 with probability

$$P(SS_2, SST_2) = P_{S_f} P_{L_f} P_t(2, 2) \quad (B.10)$$

where $P_t(2, 2)$ is defined in section B.3.2.

- if $ssthresh > 2$ then $cwnd$ is increased and TCP will go from state SS_2 to state SST_3 with probability

$$P(SS_2, SST_3) = P_{S_f} P_{L_f} P_c(3, 2). \quad (B.11)$$

where $P_c(3, 2)$ is defined in section B.3.2.

Combining Equations B.9 and B.10 yields

$$P(SS_2, SST_2) = P_{L_f} + P_{S_f} P_{L_f} P_t(2, 2) \quad (B.12)$$

Losses in SS when $3 \leq cwnd \leq W_m/4$

The probabilities $P_{TO}(j-1)$ and $P_{TD}(j-1)$ are used in the calculation of the transition probabilities $P(SS_i, SST_j)$ and $P(SS_i, SSF_j)$ for $cwnd$ values $i \geq 3$ to account for the loss patterns where $j-1$ packets are sent after the first loss.

Consider a packet that is transmitted when TCP is in state SS_i and is subsequently lost. Thus $cwnd = i$ when the lost packet was transmitted and $cwnd$ continues to grow until the transmitter detects the loss or $ssthresh$ is reached. Thus transitions from SS_i into SST_j and SSF_j occur if either the loss of a packet is detected before the $ssthresh$ is reached or if the $ssthresh$ is reached before the loss is detected.

Consider a pair of packets which are transmitted when $cwnd = i$ in SS . Let $cwnd = j$ when the loss is detected before the $ssthresh$ is reached or $ssthresh$ is reached before the loss is detected. Two cases are possible

- if $ssthresh \geq 2i$ then $j = 2i - 2$ if the first packet of the pair is lost and $j = 2i - 1$ if the second packet of the pair is lost. This is the case where the loss is detected before the $ssthresh$ is reached. In this case $i \leq ssthresh/2$ and $ssthresh \leq W_m/2$ so that $i \leq W_m/4$.
- if $ssthresh < 2i$ then $i \leq j = ssthresh < 2i$. This is the case where $ssthresh$ is reached when or before the loss is detected. Here two cases are possible
 - $ssthresh \leq 2i - 2$ and $j = ssthresh \leq W_m/2$ so that $i \geq W_m/4 + 1$
 - $ssthresh \leq 2i - 1$ and $j = ssthresh \leq W_m/2$ so that $i \geq (W_m/2 + 1)/2$.

Consider the following example which is illustrated by inspection of Figure B.3.

- Packet 6 is lost. Packet 6 is the first packet of the pair transmitted when $i = 4$. A further $i - 2$ ACKs are received namely ACK 4 and ACK 5. ACK 6 is not received within one RTT and the loss is detected by TO or TD by which time $cwnd$ has increased from i to j where $j = i + i - 2 = 2i - 2 = 6$. Thus

$$\begin{aligned} P(SS_4, SST_6) &= P_{L_f} P_c(6, 4) P_{TO}(5) \\ P(SS_4, SSF_6) &= P_{L_f} P_c(6, 4) P_{TD}(5). \end{aligned}$$

- Packet 4 is successful and packet 5 is lost. Packet 5 is the second packet of the pair transmitted when $i = 3$. A further $i - 1$ ACKs are received namely ACK 3 and ACK 4. ACK 5 is not received within one RTT and the loss is detected by TO or TD by which time $cwnd$ has increased from i to j where $j = i + i - 1 = 2i - 1 = 5$. Thus

$$\begin{aligned} P(SS_3, SST_5) &= P_{S_f} P_{L_f} P_c(5, 3) P_{TO}(4) \\ P(SS_3, SSF_5) &= P_{S_f} P_{L_f} P_c(5, 3) P_{TD}(4). \end{aligned}$$

If $i < 3$

$$P(SS_i, SST_j) = \begin{cases} P_{L_f} & i = 1 \quad j = 1 \\ P_{L_f} + P_{S_f} P_{L_f} P_t(2, 2) & i = 2 \quad j = 2 \\ P_{S_f} P_{L_f} P_c(3, 2) & i = 2 \quad j = 3 \end{cases}$$

If $i \geq 3$ and the first packet of the pair is lost the transitions due to timeout loss in SS are

$$P(SS_i, SST_j) = \begin{cases} P_{L_f} P_t(j, i) P_{TO}(j - 1) & 3 \leq i < W_m/4 + 1 \quad i \leq j < 2(i - 1) \\ P_{L_f} P_c(j, i) P_{TO}(j - 1) & 3 \leq i < W_m/4 + 1 \quad j = 2(i - 1) \\ P_{L_f} P_t(j, i) P_{TO}(j - 1) & i \geq W_m/4 + 1 \quad i \leq j \leq W_m/2 \end{cases}$$

and the transitions due to triple duplicate ACK loss in SS are

$$P(SS_i, SSF_j) = \begin{cases} P_{L_f} P_t(j, i) P_{TD}(j - 1) & 3 \leq i < W_m/4 + 1 \quad i \leq j < 2(i - 1) \\ P_{L_f} P_c(j, i) P_{TD}(j - 1) & 3 \leq i < W_m/4 + 1 \quad j = 2(i - 1) \\ P_{L_f} P_t(j, i) P_{TD}(j - 1) & i \geq W_m/4 + 1 \quad i \leq j \leq W_m/2. \end{cases}$$

If $i \geq 3$ and the second packet of the pair is lost the transitions due to timeout loss in SS are

$$P(SS_i, SST_j) = \begin{cases} P_{L_f} P_{S_f} P_t(j, i) P_{TO}(j - 1) & 3 \leq i < (W_m/2 + 1)/2 \quad i \leq j < 2i - 1 \\ P_{L_f} P_{S_f} P_c(j, i) P_{TO}(j - 1) & 3 \leq i < (W_m/2 + 1)/2 \quad j = 2i - 1 \\ P_{L_f} P_{S_f} P_t(j, i) P_{TO}(j - 1) & i \geq (W_m/2 + 1)/2 \quad i \leq j \leq W_m/2 \end{cases}$$

and the transitions due to triple duplicate ACK loss in SS are

$$P(SS_i, SSF_j) = \begin{cases} P_{L_f} P_{S_f} P_t(j, i) P_{TD}(j-1) & 3 \leq i < (W_m/2 + 1)/2 \quad i \leq j < 2i - 1 \\ P_{L_f} P_{S_f} P_c(j, i) P_{TD}(j-1) & 3 \leq i < (W_m/2 + 1)/2 \quad j = 2i - 1 \\ P_{L_f} P_{S_f} P_t(j, i) P_{TD}(j-1) & i \geq (W_m/2 + 1)/2 \quad i \leq j \leq W_m/2. \end{cases}$$

Losses in CA

A TCP connection transitions from CA_i into CAT_i or CAF_i if the first packet sent in CA_i is lost. It goes into CAT_{i+1} or CAF_{i+1} if the first packet is successfully received and any of the packets sent after the first packet is lost. This first lost packet can be any of the j , $2 \leq j \leq i + 1$ packets sent after the first packet.

The transition probability due to TO loss in CA_2 when the window size does not change is given by

$$P(CA_2, CAT_2) = P(\text{ the first packet is lost}) = P_{L_f}.$$

The transition probability due to TO loss in CA_2 when the window size does change is given by

$$\begin{aligned} P(CA_2, CAT_3) &= P(\text{ the first packet is successful and the first loss is in the } j^{th} \text{ position, } 2 \leq j \leq 3) \\ &= P_{S_f} P_{L_f} + P_{S_f}^2 P_{L_f} \\ &= P_{S_f} (1 - P_{S_f}^2). \end{aligned}$$

The transition probability due to timeout loss in CA_3 when the window size does not change is

$$P(CA_3, CAT_3) = P(\text{ the first packet is lost}) = P_{L_f}.$$

The transition probability due to timeout loss in CA when the window size does not change for $4 \leq i \leq W_m - 1$ is

$$\begin{aligned} P(CA_i, CAT_i) &= P(\text{ the first packet is lost}) \times \\ &\quad P(\text{timeout loss given that } i - 1 \text{ packets were sent after the first loss}) \\ &= P_{L_f} P_{TO}(i - 1). \end{aligned}$$

The transition probability due to timeout loss in CA when the window size changes for

$3 \leq i \leq W_m - 1$ is

$$\begin{aligned}
 & P(CA_i, CAT_{i+1}) \\
 &= P(\text{ the first packet is successful and the first loss is in the } j^{th} \text{ position, } 2 \leq j \leq i+1) \\
 &\quad \times P(\text{timeout loss given that } i \text{ packets were sent after the first loss}) \\
 &= (P_{S_f}P_{L_f} + P_{S_f}^2P_{L_f} + \dots + P_{S_f}^iP_{L_f})P_{TO}(i) \\
 &= P_{S_f}(1 - P_{S_f}^i)P_{TO}(i).
 \end{aligned}$$

The transition probability due to a triple duplicate ACK loss in CA when the window size does not change for $4 \leq i \leq W_m - 1$ is

$$\begin{aligned}
 & P(CA_i, CAF_i) \\
 &= P(\text{ the first packet is lost}) \times \\
 &\quad P(\text{triple duplicate ACK loss given that } i-1 \text{ packets were sent after the first loss}) \\
 &= P_{L_f}P_{TD}(i-1).
 \end{aligned}$$

The transition probability due to triple duplicate ACK loss in CA when the window size changes for $3 \leq i \leq W_m - 1$ is

$$\begin{aligned}
 & P(CA_i, CAF_{i+1}) \\
 &= P(\text{ the first packet is successful and the first loss is in the } j^{th} \text{ position, } 2 \leq j \leq i+1) \\
 &\quad \times P(\text{triple duplicate ACK loss given that } i \text{ packets were sent after the first loss}) \\
 &= (P_{S_f}P_{L_f} + P_{S_f}^2P_{L_f} + \dots + P_{S_f}^iP_{L_f})P_{TO}(i) \\
 &= P_{S_f}(1 - P_{S_f}^i)P_{TD}(i).
 \end{aligned}$$

When $cwnd = W_m$

$$\begin{aligned}
 P(CA_{W_m}, CAT_{W_m}) &= P_{S_f}(1 - P_{S_f}^{W_m})P_{TO}(W_m - 1) \\
 P(CA_{W_m}, CAF_{W_m}) &= P_{S_f}(1 - P_{S_f}^{W_m})P_{TD}(W_m - 1).
 \end{aligned}$$

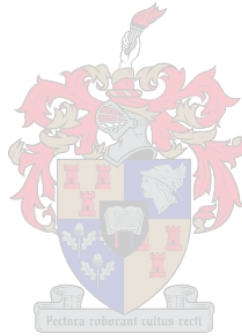
If TCP is in the SSF , SST , CAT or CAF states, it goes into the SS state where the window size is reduced to 1 to re-transmit the lost packet. If TCP is in the SST or CAT states, i.e. if the loss is a timeout loss, it tries $c = 16$ re-transmissions before it closes the connection. Here the timeout value doubles for every re-transmission loss until it becomes 64 times the original value. Closed connections are assumed to re-open and start in the SS_1 state after a time of 180s.

B.4 Chapter summary

This chapter presents a detailed state transition graph of our 6 state model which shows the steady state behavior of TCP connections. We have also presented another state transition graph which is similar to the queueing network model of Polito and which separately shows the exponential back off and the retransmission states which are built into our *SS* and *SST* states.

An idealized TCP transmission cycle which is also presented in the Polito model and which explains how TCP transitions from one state to another state is also given in this chapter.

In this chapter we investigated the Polito model and gave detailed derivation and explanation to all the formulas used in our model and in the Polito model based on the description of the TCP algorithms. Instead of these formulas other simpler transition rates and formulas can also be used in our general modeling framework with small modifications.



Appendix C

The service times of each TCP queue

The derivation of the average time $\tau(A_i)$ which TCP spends in state (queue) A when $cwnd = i$ for each state is presented below.

The derivations of the service times require the average Round Trip Time \overline{RTT} which is the time from the start of the transmission of a data packet until the corresponding ACK is received. The \overline{RTT} is obtained from our model as shown in section 6.2.2. The initial timeout value T_0 which is used in the calculation of the service times is approximated by Equation 2.3. From Figure B.3, it can be seen that $\tau(SS_1)$ and $\tau(SS_2)$ are equal to \overline{RTT} . For the other service times in SS , it is assumed that the ACKs are uniformly spaced in the interval between the two ACKs that cause the window size w to be a power of 2 ($w = 2^i$) as it is difficult to predict the spacing between the ACKs. The fact that the window size doubles every RTT in SS makes $w = 2^i$ more likely than other values as verified by simulation experiments of the $cwnd$ size distribution in section 7.5. Even though the ACKs are assumed to be uniformly distributed, it can be seen from Figure B.3 that TCP spends more time in the 2^i $cwnd$ size values than the other $cwnd$ values in the same RTT before the $ssthresh$ is reached. For example consider the third time line of Figure B.3. After the arrival of ACK 3 TCP spends most of the RTT in SS_4 . It is assumed that the time spent in SS with $cwnd = i$ is inversely proportional to i and is set to $\sigma \overline{RTT}$ when $w = 2^i$ where $\sigma \in [1/w, 1]$. The remaining part of \overline{RTT} which is $\overline{RTT} - \sigma \overline{RTT} = (1 - \sigma) \overline{RTT}$ is spent in SS when $w \neq 2^i$. There are $2^n - 1 - (2^{n-1} + 1) + 1 = 2^{n-1} - 1$, $n \geq 2$ window size values not equal to 2^i (not powers of 2). Now each SS_i state with these window size values has a service time of $\frac{(1-\sigma)\overline{RTT}}{2^{n-1}-1}$. As shown in [38], quantitative results excluding the distribution of the window size w are almost independent of the value chosen for σ in $[1/w, 1]$ where w is the window size

and we set $\sigma = 2/3$. It is shown in [38] that this choice gives satisfactory results also for the distributions of *cwnd* and *ssthresh*. Hence

$$\tau(SS_i) = \begin{cases} \overline{RTT}, & i = 1, 2 \\ \sigma \overline{RTT}, & i = 2^n, n \geq 2 \\ \frac{(1-\sigma)\overline{RTT}}{2^{n-1}-1}, & 2^{n-1} < i < 2^n, n > 1. \end{cases}$$

The derivations of the service times $\tau(SST_i)$ are based on the fact that the timeout starts in the queue where the lost packet is transmitted which can be a queue with a different *cwnd* value from the one where the timeout expires. The timeout T_0 minus the service time of the previously visited queue is used for smaller window values, $1 \leq i \leq 2$ to obtain $\tau(SST_i)$. The derivation of the service time of queue *SST* when *cwnd* = *i* in retransmission *j* ($\tau(SST_i^j)$), $0 \leq j \leq c$ follows from TCP's exponential back-off. Note that $\tau(SST_1^c) = 180s$ as we assume that closed connections re-open after 180sec. Hence

$$\tau(SST_i^j) = \begin{cases} T_0 - \overline{RTT}, & 1 \leq i \leq 3, j = 0 \\ T_0, & i \geq 4, j = 0 \\ 2^j T_0, & i = 1, 1 \leq j \leq 6 \\ 64T_0, & i = 1, 7 \leq j \leq c-1 \\ 180sec, & i = 1, j = c \\ T_0 - \overline{RTT}, & 2 \leq i \leq 3, j = 1 \\ \tau(SST_i^{j-1}) - \overline{RTT}, & i = 2, 3, 2 \leq j \leq c. \end{cases}$$

For larger window values the timeout queue is reached from different queues, and many packets are sent during the service times of the queues. Therefore $\tau(SST_i)$ is approximated by $T_0 - \overline{RTT}/2$ as it is assumed that packets which belong to large windows are dispersed uniformly along the transmission pipe. The timed-out packet is assumed to be transmitted, on average, in the middle of the RTT. Thus only half of \overline{RTT} is subtracted. Therefore

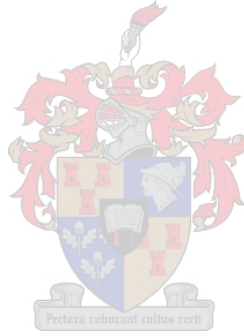
$$\tau(CAT_i) = \begin{cases} T_0 - \overline{RTT}, & i = 2 \\ T_0 - \overline{RTT}/2, & 3 \leq i \leq W_m. \end{cases}$$

The derivation of the service times $\tau(SSF_i)$ and $\tau(CAF_i)$ assume that the lost packet is transmitted on average in the middle of the *RTT*. This contributes $\overline{RTT}/2$ towards the service times of *SSF* and *CAF*. It is also assumed that the times when the ACKs of the packets sent within one *RTT* arrive are uniformly distributed in the next *RTT*. Therefore each duplicate ACK is received after \overline{RTT}/i . ACK of the lost packet was supposed to be received after \overline{RTT}/i in the second \overline{RTT} . So the first of the triple duplicate ACKs is received after $2\overline{RTT}/i$, the second after $3\overline{RTT}/i$ and the third duplicate ACK is received at $4\overline{RTT}/i$. This contributes

$4\overline{RTT}/i$ to the service times of *SSF* and *CAF*. This is the time needed for triple duplicate ACKs to be received. Hence

$$\begin{aligned}\tau(SSF_i) &= \overline{RTT}/2 + 4\overline{RTT}/i \\ \tau(CAF_i) &= \overline{RTT}/2 + 4\overline{RTT}/i.\end{aligned}$$

As can be seen from Figure B.3 $\tau(CA_i) = \overline{RTT}$.



Appendix D

The number of packets offered by each TCP queue

The derivation of the number $L(A_i)$ of packets offered from queue A_i to the underlying IP network is given below. As shown in Figures B.3 and B.4, TCP starts by sending one packet in SS and each time the window is increased by 1 in SS , TCP offers 2 packets to the network. Therefore

$$L(SS_i) = \begin{cases} 1 & \text{if } i = 1 \\ 2 & \text{otherwise.} \end{cases}$$

The number of packets transmitted by a TCP source in slow start from the loss instant to the loss detection depends on the window size w' when the loss is detected and whether the lost packet was the first or the second packet transmitted with window size i . The probability that either the first or second packet is lost in the previously visited queue is

$$P_{L_f} + P_{S_f}P_{L_f} = P_{L_f}(1 + P_{S_f}) = (1 - P_{L_f})(1 + P_{S_f}) = 1 - P_{S_f}^2.$$

The probability that the first packet sent in SS_i is lost given that there is a loss is $P_{L_f}/(1 - P_{S_f}^2)$ and the probability that the first packet is successful and the second packet is lost is $P_{S_f}P_{L_f}/(1 - P_{S_f}^2)$. As explained in section B.2, the first packet loss is detected when $w' = 2i - 2$ and the second packet loss is detected when $w' = 2i - 1$. Hence $2i - 2 - i = i - 2$ and $2i - 1 - i = i - 1$ packets are sent until the first packet and the second packet losses are detected respectively. No packets are generated in SST_1 . If the first packet sent in SS_2 is lost, then no packet is offered to the network in SST_2 . If the first packet sent in SS_2 is successful, the second packet is lost and $ssthresh$ is reached then TCP is also in SST_2 . The probability

of this event is $\frac{P_{S_f} P_{L_f} P_t(2,2)}{P_{L_f} + P_{S_f} P_{L_f} P_t(2,2)}$. Therefore

$$L(SST_i) = \begin{cases} 0 & i = 1 \\ \frac{P_{S_f} P_{L_f} P_t(2,2)}{P_{L_f} + P_{S_f} P_{L_f} P_t(2,2)}, & i = 2 \\ (i-2) \frac{P_{L_f}}{1-P_{S_f}^2} + (i-1) \frac{P_{S_f} P_{L_f}}{1-P_{S_f}^2} = L(SSF^{(i)}), & \text{otherwise} \end{cases} \quad (D.1)$$

From the above derivations it can be seen that

$$L(SST_3^j) = (3-1) \frac{P_{S_f} P_{L_f}}{P_{S_f} P_{L_f}} = 2, \quad 1 \leq j \leq c-1$$

as the only transition into SST_3^j happens when the second packet sent in SS_2^j which models Karn's algorithm is lost. As shown in section B.2 the number of packets transmitted in CA_i if the first packet is successfully transmitted is $i+1$ and is i if the first packet sent in CA_i is lost. Therefore

$$L(CA_i) = \begin{cases} P_{L_f} i + (1 - P_{L_f})(i+1) & 2 \leq i \leq W_m - 1 \\ W_m & \text{otherwise} \end{cases} \quad (D.2)$$

The calculation of the load offered to the underlying IP network by the queues CAF and CAT is

$$\Lambda_{FTO} = \sum_{i=2}^W \left(\sum_{j=1}^i j P_{S_f}^j P_{L_f} \right) \lambda(CA_i). \quad (D.3)$$

This load offered by the respective queues depends on which packet was lost among the packets previously transmitted in CA_i .

The $\lambda(A_i)$ are obtained from Equation 6.17. Instead of Equation 6.18, the following equation can be used in the calculation of Λ .

$$\Lambda = \sum_{i=1}^{W_m} i \lambda^*(i) \quad (D.4)$$

where the rate $\lambda^*(i)$ at which batches of size i are sent into the network is given by

$$\lambda^*(i) = \sum_{A \in \tau} \sum_{L(A_k)=i} \lambda(A_k). \quad (D.5)$$

The $\lambda^*(i)$ can also be splitted into $\lambda_b(i)$ which denotes the arrival rate if i packets are sent before congestion (before a loss occurs) and $\lambda_a(i)$ which denotes the arrival rate if i packets are sent after congestion.

Calculation of $\lambda_b(i)$

The arrival rate $\lambda_b(1)$ if 1 packet is offered before congestion is equal to the arrival rate $\lambda(SS_1)$.

Two packets are offered to the network before congestion if TCP is in any of the SS_i , $2 \leq i \leq W_m/2$ states or if the first packet sent in CA_2 is lost as shown in Equation D.2. Therefore

$$\lambda_b(2) = \sum_{j=2}^{W_m/2} \lambda(SS_j) + P_{L_f} \lambda(CA_2). \quad (D.6)$$

The number of packets offered by TCP connections to the network before congestion is i , $3 \leq i \leq W_m - 1$ if either the first packet sent in the CA_i is lost or the first packet sent in the CA_{i-1} state is successful as shown in Equation D.2. The number of packets offered before congestion is W_m if TCP is in the CA_{W_m} state implying that the arrival rate $\lambda_b(W_m)$ if W_m packets are offered to the network before congestion is equal to the arrival rate of the CA_{W_m} queue. Therefore

$$\lambda_b(i) = \begin{cases} P_{L_f} \lambda(CA_i) + (1 - P_{L_f}) \lambda(CA_{i-1}) & 3 \leq i \leq W_m - 1 \\ \lambda(CA_i) & i = W_m. \end{cases} \quad (D.7)$$

Calculation of $\lambda_a(i)$

As shown in Equation D.1 above no packet is offered by the SST_1 state. If the first packet sent in SS_2 is lost then TCP goes to the SST_2 state where no $(2 - 2 = 0)$ packet is offered to the network as explained at the beginning of this section. Similarly if the first packet sent in CA_i is lost TCP goes to the CAT_i state to wait for a timeout to expire and hence no other packet is sent in CAT_i . The arrival rate $\lambda_a(0)$ is therefore given by

$$\lambda_a(0) = \lambda(SST_1) + P_{L_f} \lambda(SS_2) + \sum_{j=2}^{W_m} P_{L_f} \lambda(CA_j). \quad (D.8)$$

Only one packet is offered to the network after congestion if

- the first of the two packets sent in SS_2 is successful and the second is lost and *ssthresh* is reached when the *cwnd* = 2. As explained above the probability of this event is $P_{S_f} P_{L_f} P_t(2, 2)$.
- the first packet sent in SS_3 is lost and TCP is in the SST_3 state where $3 - 2 = 1$ packet is sent until the first loss is detected. The probability of this event is $P_{L_f} / (1 - P_{S_f}^2)$.
- the first packet sent in CA_i is successful and the second packet sent in CA_i is lost in which case TCP is in the CAT_{i+1} or CAF_{i+1} states. As shown in Equation D.3 the probability of this event is $P_{S_f} P_{L_f}$.

Therefore

$$\lambda_a(1) = P_{S_f} P_{L_f} P_t(2, 2) \lambda(SS_2) + \frac{P_{L_f}}{1 - P_{S_f}^2} \lambda(SST_3) + \sum_{j=2}^{W_m} P_{S_f} P_{L_f} \lambda(CA_j). \quad (D.9)$$

Two packets are offered to the network after congestion if

- the first two packets sent CA_{i-1} are successful and the third packet is lost. The probability of this event is $P_{S_f}^2 P_{L_f}$.
- the first packet sent in SS_4 is lost and TCP is in the SST_4 or SSF_4 states where each send $4 - 2 = 2$ packets until the first loss is detected. The probability of this event is $P_{L_f}/(1 - P_{S_f}^2)$.
- the first packet sent in SS_2 is successful and the second packet is lost in which case TCP is in the SST_3 state where $3 - 1 = 2$ packets are sent until the loss is detected as shown in Equation D.1. The probability of this event is $P_{S_f} P_{L_f}/(1 - P_{S_f}^2)$.

Therefore

$$\lambda_a(2) = \sum_{j=3}^{W_m} P_{S_f}^2 P_{L_f} \lambda(CA_{j-1}) + \frac{P_{L_f}}{1 - P_{S_f}^2} (\lambda(SST_4) + \lambda(SSF_4)) + \frac{P_{S_f} P_{L_f}}{1 - P_{S_f}^2} \lambda(SST_3). \quad (D.10)$$

The number of packets offered to the network after congestion is i , $3 \leq i \leq W_m/2 - 2$ if

- the first i of the packets sent CA_{j-1} , $i + 1 \leq j \leq W_m$ are successful and the $i + 1$ the packet is lost. The probability of this event is $P_{S_f}^i P_{L_f}$.
- TCP is in either the SST_{i+2} or SSF_{i+2} states where each send $i + 2 - 2 = i$ packets until the loss of the first packet is detected. The probability of this event is $P_{L_f}/(1 - P_{S_f}^2)$.
- TCP is in either the SST_{i+1} or SSF_{i+1} states where each send $i + 1 - 1 = i$ packets until loss of the second packet is detected. The probability of this event is $P_{S_f} P_{L_f}/(1 - P_{S_f}^2)$ as shown in Equation D.1.

Therefore for $3 \leq i \leq W_m/2 - 2$

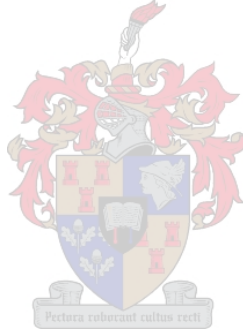
$$\lambda_a(i) = \sum_{j=i+1}^{W_m} P_{S_f}^i P_{L_f} \lambda(CA_{j-1}) + \frac{P_{L_f}}{1 - P_{S_f}^2} (\lambda(SST_{i+2}) + \lambda(SSF_{i+2})) + \frac{P_{S_f} P_{L_f}}{1 - P_{S_f}^2} (\lambda(SST_{i+1}) + \lambda(SSF_{i+1})). \quad (D.11)$$

For $i = W_m/2 - 1$

$$\lambda_a(i) = \sum_{j=i+1}^{W_m} P_{S_f}^i P_{L_f} \lambda(CA_{j-1}) + \frac{P_{S_f} P_{L_f}}{1 - P_{S_f}^2} (\lambda(SST_{i+1}) + \lambda(SSF_{i+1})) \quad (\text{D.12})$$

and for $W_m/2 \leq i \leq W_m - 1$

$$\lambda_a(i) = \sum_{j=i+1}^{W_m} P_{S_f}^i P_{L_f} \lambda(CA_{j-1}). \quad (\text{D.13})$$



Appendix E

More models of TCP

In Chapter 6, the *cwnd* and *ssthresh* distributions are calculated after the loads offered by each queue to the network sub-model are obtained. The timeout and fast-retransmit probability formulas can also not be calculated before the loads offered by each queue are obtained. As shown in section 6.1 to calculate the loads offered involves many steps. This appendix contains simple closed form alternative models to calculate the *cwnd* and *ssthresh* distributions and the TO and TD loss probabilities directly from the transition probability matrix or other closed form formulas. These models can also be used to obtain other TCP performance metrics.

This appendix is organized as follows. We first present the TCP *cwnd* size distributions. In section E.2 we give TCP timeout and fast-retransmit probabilities. The TCP *ssthresh* distributions are formulated in section E.3. The stationary probabilities of each TCP state are presented in section E.4 and the chapter summary is given in section E.5.

E.1 The TCP *cwnd* size distribution

The TCP *cwnd* size distributions which are the basic and key steps in obtaining TCP performance metrics can be derived as follows.

Given the transition probabilities $P(SS_i, SS_{i+1})$, $P(SS_i, CA_i)$ and $P(CA_i, CA_{i+1})$ described in appendix B.3 we first present the expressions and symbols needed for the derivation of the *cwnd* size distribution.

Let $P_{SS_s}(i)$ denote the probability of successful increment of the window size i by 1 in SS

$$P_{SS_s}(i) = P(SS_i, SS_{i+1}).$$

Let $P_{SS_t}(i)$ denote the probability of failure to increase the window size i by 1 in SS because the *ssthresh* is reached

$$P_{SS_t}(i) = P(SS_i, CA_i).$$

Let $P_{SS_l}(i)$ denote the probability of failure to increase the window size, i by 1 in SS due to packet loss

$$P_{SS_l}(i) = \begin{cases} 1 - P_{SS_s}(i), & i = 1 \\ 1 - P_{SS_s}(i) - P_{SS_t}(i), & 2 \leq i < W_m/2 \\ 1 - P_{SS_t}(i) & i = W_m/2. \end{cases}$$

Let $P_{CA_s}(i)$ denote the probability of successful increment of the window size, i by 1 in CA

$$P_{CA_s}(i) = P(CA_i, CA_{i+1}).$$

Let $P_{CA_l}(i)$ denote the probability of failure to increase the window size, i by 1 in CA

$$P_{CA_l}(i) = \begin{cases} 1, & i = W_m \\ 1 - P(CA_i, CA_{i+1}), & 2 \leq i \leq W_m - 1. \end{cases}$$

Based on the bounded geometric distribution (see section 6.1.2), the two *cwnd* size distributions excluding and including the *cwnd* values where TCP waits for triple-duplicate ACKs or timeout to expire are given below. Similar distributions can be derived using the truncated geometric distribution explained in chapter 6.

E.1.1 The *cwnd*₁ model

The *cwnd* size distribution *cwnd*₁ which excludes the *cwnd* values where TCP waits for triple-duplicate ACKs and timeout to expire is obtained by conditioning on the *ssthresh* values.

Let $P_w(w | i) = P(cwnd = w | ssthresh = i)$, $2 \leq i \leq W_m/2$, $i < w$. This is given by

$$P_w(w | i) = \begin{cases} \prod_{j=1}^{w-1} P_{SS_s}(j) P_{SS_l}(w), & \text{if } w < i \\ \prod_{j=1}^{w-1} P_{SS_s}(j) P_{SS_t}(i) P_{CA_l}(i), & \text{if } i = w \\ \prod_{j=1}^{i-1} P_{SS_s}(j) P_{SS_t}(i) \prod_{k=i}^{w-1} P_{CA_s}(k) P_{CA_l}(w), & \text{if } w > i \\ P_{SS_l}(w), & \text{if } w = 1 \end{cases} \quad (\text{E.1})$$

where $i > w$ means the *ssthresh* i is not reached when the *cwnd* = w .

Using the Law of Total Probability (see Theorem 4.3.1) the probability that the window size is w , $P_w(w)$ is given by

$$P_w(w) = \sum_{i=2}^{W_m/2} P_w(w | i) P_t(i), \quad 1 \leq w \leq W_m \quad (\text{E.2})$$

where $P_t(i)$ as shown in Equation B.3 is the probability that $ssthresh = i$.

E.1.2 The $cwnd_2$ model

The $cwnd$ distribution $cwnd_2$ including the $cwnd$ values where TCP waits for triple-duplicate ACKs and timeout to expire in SS can now be obtained by replacing $P_{SS_s}(i) = P_{S_f}^2 P_c(i+1, i)$ and $P_{SS_t}(i) = P_{S_f}^2 P_t(i, i)$ with $P_{S_f} P_c(i+1, i)$ and $P_{S_f} P_t(i, i)$ respectively. The expression $P_{S_f} P_c(i+1, i)$ represents the probability that the first of the two packets sent in SS_i is successful and $ssthresh$ is not reached. The second expression, $P_{S_f} P_t(i, i)$ represents the probability that the first of the two packets sent in SS_i is successful and $ssthresh$ is reached. In this case the second packet sent in SS_i can be successful in which case TCP goes to SS_{i+1} or CA_i or the packet can be lost in which case TCP has to wait for either TD or the TO to expire when the $cwnd = i+1$. To account for the $cwnd$ size values where the first packet sent in CA_w is lost transitions from CA_w into CAT_{w+1} and CAF_{w+1} are also considered.

We next present the derivation of the TCP timeout and fast retransmit probabilities.

E.2 The TCP timeout and fast-retransmit probabilities

Given the packet loss and success probabilities, P_{L_f} , P_{L_a} , P_{S_f} and P_{S_a} discussed in section 6.1.2 we next present the timeout (TO) and triple duplicate ACK (TD) loss probabilities. As shown in Equation B.1 of the appendix, for transitions that are consequences of losses, the probability that a loss is detected by timeout given that i segments were transmitted after the lost one is

$$P_{TO}(i) = \sum_{j=0}^2 \binom{i}{j} P_{L_a}^{i-j} P_{S_a}^j$$

for $3 \leq i \leq W_m$ and the corresponding probability that a loss is detected by TD is

$$P_{TD}(i) = 1 - P_{TO}(i).$$

Now we have the following TO and TD loss probability models. Let $P(i)$ be the probability that i , $0 \leq i \leq W_m - 1$ packets are sent after the lost one. For long lived TCP connections if i packets are sent after the lost packet the $cwnd$ when these i packets are sent is equal to $i+1$. This is because full window of packets are sent every RTT . The timeout and fast retransmit models are obtained by approximating the probability that i packets are transmitted after the lost one $P(i)$ with $P(cwnd = i+1)P_{L_f} = P_w(i+1)P_{L_f}$. Instead of this rough approximation

the formula explained in [48] can be used with small modifications. Now the probability that a loss is due to TO is given by

$$P_{\text{TO}'} = \sum_{i=0}^{W_m-1} P_{\text{TO}}(i)P(i) \quad (\text{E.3})$$

where $P_{\text{TO}}(i) = 1$ for $i = 0, 1$ or 2 . The probability that TCP is in fast-retransmit $P_{\text{TD}'}$ which is also the probability that a loss is a TD loss is given by

$$P_{\text{TD}'} = \sum_{i=3}^{W_m-1} P_{\text{TD}}(i)P(i). \quad (\text{E.4})$$

Using the *cwnd* size distributions, timeout and fast-retransmit probabilities we next derive the two *ssthresh* size distributions which can be used in the next iteration of the fixed point procedure.

E.3 The *ssthresh* models

E.3.1 The *ssthresh*₁ model

The *ssthresh*₁ model gives $P_t(i)$ the probability that the *ssthresh* = i using the Bayes' rule of probability (see section 4.3.2). Let $P_t(i | w) = P(\textit{ssthresh} = i | \textit{cwnd} = w)$, $2 \leq i \leq W_m/2$. Using the Bayes' rule of probability,

$$P_t(i | w) = \frac{P_w(w | i)P_t(i)}{P_w(w)} \quad (\text{E.5})$$

where $P_t(i)$ is obtained from the previous iteration as in Equation E.2. Then the probability that the *ssthresh* = i for the next iteration is given by

$$P_t(i) = \sum_{w=1}^{W_m} P_t(i | w)P_w(w). \quad (\text{E.6})$$

E.3.2 The *ssthresh*₂ model

As explained in section B.3.2 the *ssthresh* is halved every time a loss is detected. The loss can be detected by timeout (TO) or triple duplicate ACK s (TD). Let $P_{\text{TO}'}(w)$ and $P_{\text{TD}'}(w)$ be the probabilities that a timeout and triple-duplicate ACK losses are detected respectively when the *cwnd* = w . For long lived TCP connections, a loss is detected when the *cwnd* = w means that $w - 1$ packets are sent after the lost packet. This is because the window slides sending

an additional packet for each of the ACKs received before the loss is detected. Therefore we approximate $P_{TO'}(w)$ and $P_{TD'}(w)$ as

$$\begin{aligned} P_{TO'}(w) &= P_{L_f} P_{TO}(w-1) \\ P_{TD'}(w) &= P_{L_f} P_{TD}(w-1). \end{aligned}$$

Instead of these rough approximations the timeout formulas given in Equation 5.5 and [48] can be used with small modifications to account for packet loss correlation. Now the *ssthresh* distribution is given as follows.

$$P_t(i) = \begin{cases} \sum_{j=1}^5 (P_{TO'}(j) + P_{TD'}(j)) P_w(j), & i = 2 \\ (P_{TO'}(2i) + P_{TD'}(2i)) P_w(2i) + (P_{TO'}(2i+1) + P_{TD'}(2i+1)) P_w(2i+1), & 3 \leq i \leq W_m/2 \end{cases} \quad (E.7)$$

where $P_w(i)$ is the probability that the *cwnd* size is i .

Based on the above models we next present the derivation of the stationary probabilities of the TCP states at each *cwnd* value.

E.4 The stationary probabilities of each TCP state

To find the long-term probabilities that TCP is in a certain state with the corresponding index, we first derive formulas to find the probability that TCP is in SS_1 . We next find new transition probabilities. Then we use a recursive formula to find the other stationary probabilities.

As shown in section 6.1.3 let $\pi(A)$ and $\pi(A_i)$ be the probabilities that TCP is in state A and A_i (state A when the *cwnd* = i) respectively. These probabilities are also the relative visit counts to A and A_i respectively.

Let $\pi(SS_i^j)$, $1 \leq i \leq 2$, $0 \leq j \leq c$ denote the probability that TCP is in the j th *SS* re-transmission state when the *cwnd* = i .

Consider a state A_i^j where $A \in \mathcal{T}$ and $a_{min} \leq i \leq a_{max}$, i denotes the *cwnd* size and j denotes the number of consecutive retransmission such that $0 \leq j \leq c$ for the *SS* and *SST* states and $j = 0$ for all other states. The index j is omitted for the states other than the *SS* and *SST* states. The values of a_{min} and a_{max} are given in Table 6.1. The maximum window size W_m is 64 and the maximum number of consecutive retransmissions, c before closing the connections is 16 for TCP-Tahoe as given in section 6.1.2.

E.4.1 The probability that TCP is in the SS_1

TCP (TCP-Tahoe) goes to SS_1 , the slow start state when the $cwnd = 1$, if (1) triple duplicate ACK (TD) loss occurs, (2) timeout (TO) loss occurs when TCP is transmitting a packet or retransmitting a lost packet for the j th, $1 \leq j \leq c$ time or (3) TCP re-opens connections which were closed for an excessive number, c of timeouts. Now the probability that TCP is in SS when the $cwnd = 1$ is given by,

$$\pi(SS_1) = \pi(SS_1^0) + \sum_{j=1}^c \pi(SS_1^j) + \pi(SST_1^c) \quad (E.8)$$

where $\pi(SS_1^0) = P_{TD}$, which is the probability that triple duplicate ACK losses occur. This is given by Equation E.4. This is because when triple duplicate ACK losses occur TCP (TCP-Tahoe) continues transmission in SS reducing the window size to 1. $\pi(SS_1^j)$ is the probability that TCP is in the j th SS re-transmission state when the $cwnd = 1$.

$$\pi(SS_1^1) = P_{TO}$$

where P_{TO} is the probability that transmission timeout occurs as given in Equation (E.3). TCP goes into SS_1^j , $2 < j \leq c$ if the $(j-1)$ th retransmission is not successful or if TCP is in one of the retransmission timeout states SST_2^j and SST_3^j of retransmission j as shown in Figure B.2. Therefore

$$\begin{aligned} \pi(SS_1^j) &= P_{L_f} \pi(SS_1^{j-1}) + \pi(SST_2^j) + \pi(SST_3^j) \\ &= P_{L_f} \pi(SS_1^{j-1}) + \pi(SS_2^j) (P(SS_2^j, SST_2^j) + P(SS_2^j, SST_3^j)) \\ &= P_{L_f} \pi(SS_1^{j-1}) + P_{S_f} \pi(SS_1^j) (P(SS_2^j, SST_2^j) + P(SS_2^j, SST_3^j)) \\ \implies \pi(SS_1^j) &= \frac{P_{L_f} \pi(SS_1^{j-1})}{1 - P_{S_f} (P(SS_2^j, SST_2^j) + P(SS_2^j, SST_3^j))}, \quad 1 \leq j \leq c-1 \end{aligned}$$

The probabilities $P(SS_2^j, SST_2^j)$ and $P(SS_2^j, SST_3^j)$ are respectively equal to the probabilities $P(SS_2, SST_2)$ and $P(SS_2, SST_3)$ explained in appendix B.3.

The $\pi(SST_1^c)$ is the probability that TCP is in the SST_1 state and c^{th} retransmission.

$$\pi(SST_1^c) = P_{L_f} \pi(SS_1^c)$$

where $\pi(SS_1^c)$ is the probability that the c th retransmission when the window size is 1 times out. The $\pi(SST_1^c)$ is the probability that TCP connections were closed for c consecutive retransmissions.

We also have that

$$\pi(SS_i) = \sum_{j=0}^c \pi(SS_i^j), \quad i = 1, 2.$$

Now

$$P(SS_i, SST_k^j) = \frac{\pi(SS_i^j)}{\pi(SS_i)} P(SS_i, SST_k) \quad (\text{E.9})$$

where $1 \leq i \leq 2$, $k = 1$ for $i = 1$, $k = 2, 3$ for $i = 2$ and $0 \leq j \leq c$.

$P(SST_1^j, SS_1) = P(SST_2^j, SS_1) = P(SST_3^j, SS_1) = 1$ as TCP goes back to SS_1 when a (timeout) loss occurs.

The other $P(A_i, B_j)$'s used in the derivation of the stationary probabilities of the TCP states are given in appendix B.3.

Now that we have the probability that TCP is in SS when the $cwnd = 1$ and the transition probabilities as given above, we use the following recursive formula to find all other stationary probabilities that TCP is in a given state.

$$\pi(A_i^j) = \sum_{B \in \mathcal{T}} \sum_{k=a_{min}}^{a_{max}} \pi(B_k^j) P(B_k^j, A_i^j). \quad (\text{E.10})$$

where $A \in \mathcal{T}$. The closure

$$\sum_{A \in \mathcal{T}} \sum_{j=0}^c \sum_{i=a_{min}}^{a_{max}} \pi(A_i^j) = 1$$

is used to normalize the stationary probabilities.

Now the load generated by the TCP sub-model and the estimates of the average loss probability and RTT can be obtained using the methods described in Chapter 6.

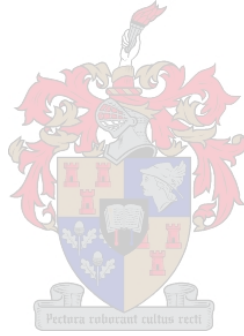
E.5 Chapter summary

This chapter presents

- models of the TCP $cwnd$ size distribution when the $cwnd$ values where TCP waits for TO or TD are included and excluded by conditioning on the $ssthresh$ probabilities based on the generalized bounded geometric distribution,
- models of the TCP TO and TD probabilities,
- two $ssthresh$ models, the first of which is obtained by conditioning on the $cwnd$ size probabilities and the second of which is obtained by using the TO, TD loss and $cwnd$ probabilities,

- closed form expressions for the probabilities that TCP is in the SS transmission state when the $cwnd = 1$ (SS_1^0) and the first SS retransmission state when the $cwnd = 1$ (SS_1^1) and
- an iterative formula where the probabilities that TCP is in a certain transmission and retransmission state when the $cwnd$ assumes any appropriate value.

Further studies can be made to build more efficient TCP models using the $cwnd$ size distributions presented in section E.1, the timeout and fast retransmit probabilities given in section E.2 and the $ssthresh$ distributions given in section E.3.1 above. For such models the service times of each state and the loads offered by each state may not be necessary. For instance they can be used with the Markov process model [42] and Markov chain model [56] for a hybrid space–terrestrial networks where the states are represented by a set of window size and $ssthresh$ values.



Appendix F

An Eritrean network topology

To further validate our analytical models, a network topology describing the path followed by Internet connections from the Ministry of Education (MOE) LAN in Eritrea [24] to web sites in the USA was also considered. The data concerning these Internet connections was collected from the Telecommunication Service of Eritrea (TSE) in the year 2003. The numerical results were similar to those obtained from the European network topology and are not reported here.

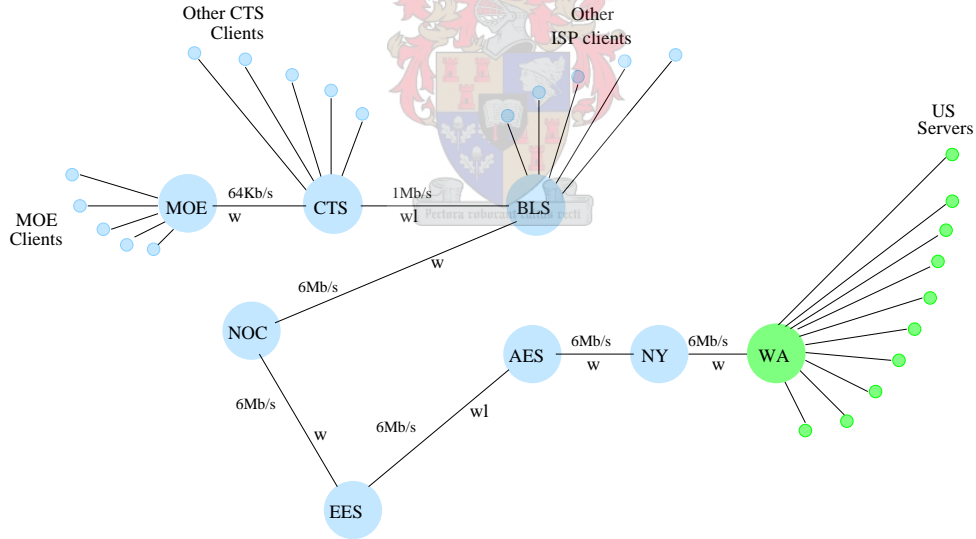


Figure F.1: The path followed by Internet connections from the MOE clients to servers in the USA

The network topology is given in Figure F.1. Computer Technology Service (CTS) is an Internet Service Provider (ISP) for the Ministry of Education (MOE). BLS is the Beitgiorgish Line of Sight to which all Eritrean ISPs are connected. The BLS is connected to the National Network Operating Center (NOC) via an optical fiber link. NOC is connected to the Eritrean

Earth Station (EES) via an optical fiber link. The EES exchanges data with the Amsterdam (The Netherlands) Earth Station (AES) via a satellite link. An undersea channel connects the router in Amsterdam with the router in New York (NY). This router in NY connects the Eritrean Internet connections with the Warsun (WA) router in USA. Warsun is the ISP of all Internet connections in Eritrea. CTS has many other customers.

In this study we are interested in TCP connections for the transfer of web pages from the US servers (web sites) to MOE clients. The distance of the MOE clients from the MOE router is assumed to be uniformly distributed between 0.005 and 15 km. The MOE router is connected to the CTS router via a 64Kb/s leased line which is some 2.5 km long. The CTS router is connected to the router at BLS via a wireless (wl) link which is some 0.5 km long and which carries a maximum of 1Mb/s. As the maximum bandwidth amount for Internet in Eritrea is only 6Mb/s, all of the links connecting the BLS with the WA ISP in the USA transfer data at a maximum rate of 6Mb/s. The links however have enough capacity should the Eritrean government buys more bandwidth.

The links labelled "w" are wired and those labelled "wl" are wireless. Wired connections can loss packets only in the bottleneck, while there are further losses in the wireless links. To model these wireless losses, a Bernoulli Loss Process with Two State Gilbert Wireless Model given in [35] can be incorporated. But in this study we analyze only the wired links as the potential capacity of all links is big enough that we assume losses due to types of links are negligible.

From the perspective of MOE and Telecommunication Service of Eritrea (TSE), one of the problems which can be addressed by the analytic models is as follows. How much bandwidth, what capacity router and associated facilities or equipments should the MOE and TSE buy in order to guarantee a certain Internet Quality of Service (QoS) level and accommodate a certain number of customers cost effectively? The QoS and other planning and dimensioning parameters are accounted for by the different performance metrics obtained from the analytical models described in this thesis.

Bibliography

- [1] E. Alessio, M. Garetto, R. Lo Cigno, M. Meo, and M. A. Marsan. Analytical estimation of completion times of mixed New-Reno and Tahoe TCP connections over single and multiple bottleneck networks. In *IEEE Globecom*, San Antonio, Texas, USA, November 2001.
- [2] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of TCP/IP with stationary random losses. *ACM SIGCOMM Computer Communication Review*, 30:231–242, 2000.
- [3] E. Altman, T. Jiménez, and R. Nunez-Queija. Analysis of two competing TCP/IP connections. *Performance Evaluation*, 49(1):43–55, September 2002.
- [4] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *SIGCOMM 04*, Portland, Oregon, USA, August 2004.
- [5] Å. Arvidsson and A. E. Krzesinski. The design of optimal multi-service MPLS networks. In *Proceedings of 10th International Telecommunication Network Strategy and Planning Symposium*, pages 31–40, June 2001.
- [6] Å. Arvidsson and A. E. Krzesinski. A model of a TCP link. In *15th ITC Specialist Seminar on Internet Traffic Engineering and Traffic Management*, Wurzburg, Germany, July 2002.
- [7] S. Athuraliya and S. Low. Optimization flow control - II: Implementation. Technical report, Department of EEE, University of Melbourne, Australia, 2000.
- [8] F. Baccelli and D. Hong. Flow level simulation of large IP networks. In *Proceedings of INFOCOM 2003*, San Francisco, California, USA, April 2003.
- [9] F. Baccelli and D. Hong. Interaction of TCP flows as billiards. In *Proceedings of INFOCOM 2003*, San Francisco, California, USA, April 2003.

- [10] F. Baccelli and D. McDonald. A square root formula for the rate of non-persistent TCP flows. In *First Conference on Next Generation Internet Networks (NGI 2005)*, pages 171–176, Rome, Italy, April 2005.
- [11] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation*. Prentice Hall International Series in Industrial and Systems Engineering, Prentice Hall, 2001.
- [12] C. Barakat and E. Altman. Performance of short TCP transfers. In *Proceedings of Networking 2000*, Paris, France, May 2000.
- [13] J. V. L. Beckers, I. Hendrawan, R. E. Kooij, and R. D. van der Mei. Generalized processor sharing performance models for Internet access lines. In *Proceedings of 9th IFIP Conference on Performance Modeling and Evaluation of ATM & IP Networks*, pages 101–112, Budapest, June 2001.
- [14] G. Bolsh, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains. Modeling and Performance Evaluation with Computer Science Applications*. A Wiley-Inter science publication, John Wiley and Sons Inc., NY 10158-0012, USA, 1998.
- [15] T. Bonald. Comparison of TCP Reno and TCP Vegas via fluid approximation. Technical report, Technical Report RR-3563, INRIA, France, 1998.
- [16] T. Bonald, A. Proutiere, G. Régnié, and J. Roberts. Insensitivity results in statistical bandwidth sharing. In *Proceedings of Seventeenth International Teletraffic Congress, ITC'17*, Salvador da Bahia, Brazil, December 2001.
- [17] Kim C. Border. *Fixed Point Theorems with Applications to Economics and Game Theory*. Press Syndicate of the University of Cambridge, The Pitt Building, Trumpington Street, Cambridge, United Kingdom, 1985.
- [18] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In *ACM SIGCOMM Conference*, pages 24–35, London, UK, September 1994.
- [19] L. S. Brakmo and L. L. Peterson. TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, 1995.

- [20] T. Bu and D. Towsley. Fixed point approximations for TCP behavior in an AQM network. In *Proceedings of ACM SIGMETRICS*, Cambridge, Massachusetts, USA, June 2001.
- [21] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *Proceedings of the 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM 2000)*, Tel Aviv, Israel, March 2000.
- [22] C. Casetti and M. Meo. A new approach to model the stationary behavior of TCP connections. In *INFOCOM 2000*, Tel Aviv, Israel, March 2000.
- [23] C. Casetti and M. Meo. An analytical framework for the performance evaluation of TCP Reno connections. *Computer Networks*, 37(5):669–687, November 2001.
- [24] CIA. *Eritrea*. The World Fact Book, <http://www.odci.gov/cia/publications/factbook/geos/er.html>.
- [25] R. Lo Cigno and M. Gerla. Modeling window based congestion control protocols with many flows. *Performance Evaluation*, 36:289–306, August 1999.
- [26] D. E. Comer. *Internetworking with TCP/IP – Principles, Protocols, and Architectures, 4th Edition*. Prentice Hall, New Jersey, 07458, USA, 2000.
- [27] P. Dimopoulos, P. Zeephongsekul, and Z. Tari. Modeling the burstiness of the TCP protocol. In *the 12th Annual Meeting of the IEEE / ACM International Symposium in Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Volendam, The Netherlands, October 2004.
- [28] G.J. Etgen. *Calculus of One and Several Variables, 8th Edition*. John Wiley & Sons Inc., New York, NY 10158-0012, USA, 1999.
- [29] K. Fall and S. Floyd. Simulation-based comparisons of Tahoe, Reno, and SACK TCP. *Computer Communication Review*, 26(3):5–21, July 1996.
- [30] P. J. S. G. Ferreira. *Fixed point problems: an introduction*. REVISTA DO DETUA, Universidade de Aveiro, Portugal, 1996.
- [31] Debessay Fesehaye and A. E. Krzesinski. A fast and accurate analytical model of TCP performance. In *Proceedings of the Southern African Telecommunication Networks and Applications Conference (SATNAC 2005)*, Central Drakensberg, KwaZulu Natal, South Africa, September 2005.

- [32] Debessay Fesehayee and A. E. Krzesinski. A queueing network model of TCP performance. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists (SAICSIT 2005)*, to appear in *ACM International Conference Proceedings Series*, White River, South Africa, September 2005.
- [33] S. Floyd. The New-Reno modification to TCP's fast recovery algorithm. In *IETF RFC 2582*, April 1999.
- [34] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.
- [35] R. Fracchia, M. Garetto, and R. Lo Cigno. A queueing network model of short-lived wired and wireless access links. In *IEEE QoSIP 2003 Workshop*, Milano, Italy, February 2003.
- [36] S. B. Fredj, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts. Statistical bandwidth sharing: a study of congestion at flow level. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 111–122, San Diego, California, USA, 2001.
- [37] M. Garetto, R. Lo Cigno, M. Meo, E. Alessio, and M. A. Marsan. Modeling short-lived TCP connections with open multi class queueing networks. In *Proceedings of 7th International Workshop on Protocols For High-Speed Networks (PFHSN)*, Berlin, Germany, April 2002.
- [38] M. Garetto, R. Lo Cigno, M. Meo, and M. A. Marsan. A detailed and accurate closed queueing network model of many interacting TCP flows. In *INFOCOM 2001*, pages 1706–1715, Anchorage, Alaska, April 2001.
- [39] M. Garetto, R. Lo Cigno, M. Meo, and M. A. Marsan. Closed queueing network models of interacting long-lived TCP flows. *IEEE/ACM Transactions on Networking*, 12(2):300–311, April 2004.
- [40] M. Garetto, R. Lo Cigno, M. Meo, and M. A. Marsan. Modeling short-lived TCP connections with open multi-class queueing networks. *Computer Networks*, 44(2):153–176, February 2004.
- [41] R. Gibbens, S. Sargood, C. Van Eijl, F. Kelly, H. Azmoodeh, R. Macfadyen, and N. Macfadyen. Fixed-point models for the end-to-end performance analysis of IP networks. In *Proceedings of 13th ITC Special Seminar: IP Traffic Management, Modeling and Management*, Monterey, California, September 2000.

- [42] G. Hasegawa and M. Murata. Analysis of dynamic behaviors of many TCP connections sharing Tail-Drop/RED routers. In *Proceedings of IEEE GLOBECOM 2001*, San Antonio, Texas, USA, November 2001.
- [43] J. J. Herings, G. van der Laan, D. Talman, and Z. Yang. A fixed point theorem for discontinuous functions. In *Tinbergen Institute Discussion Papers 05-004/1, Department of Econometrics and Tinbergen Institute*, Vrije Universiteit, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands, December 2004.
- [44] C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong. A control theoretic analysis of RED. In *INFOCOM*, pages 1510–1519, 2001.
- [45] V. Jacobson. Congestion avoidance and control. In *Proceedings of SIGCOMM '88*, Stanford, CA, August 1988.
- [46] H. Jiang and C. Dovrolis. The origin of TCP traffic burstiness in short time scales. Technical report, Gatech CERCS Technical Report, GIT-CERCS-04-09, February 2004.
- [47] I. Kaj and J. Olsén. Throughput modeling and simulation for single connection TCP-Tahoe. In *The 17th International Teletraffic Congress (ITC-17)*, Salvador da Bahia, Brazil, December 2001.
- [48] I. Kaj and J. Olsén. Slow start window modeling for single connection TCP-Tahoe. Technical report, *U.U.D.M. Report 2001:21*, Uppsala University, July 2002.
- [49] I. Kaj and J. Olsén. Stochastic equilibrium modeling of the TCP dynamics in various AQM environments. In *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'02)*, San Diego, USA, July 2002.
- [50] C. T. Kelly. *Engineering flow controls for the Internet, A dissertation submitted to the Cambridge University for the degree of Doctor of Philosophy*. PhD thesis, LABORATORY FOR COMMUNICATION ENGINEERING, Department of Engineering, Cambridge University, 2004.
- [51] F. Kelly. Mathematical modeling of the Internet. In *Bjorn Engquist and Wilfried Schmid (Eds.), Mathematics Unlimited – 2001 and Beyond*, 2001.
- [52] A. E. Krzesinski. *Lecture Notes on Teletraffic Optimization Course*. Department of Computer Science, University of Stellenbosch, South Africa, 2002.

- [53] A. Kumar. Comparative performance analysis of versions of TCP in a local network with a lossy link. *IEEE/ACM Transactions on Networking*, 6(4):485–498, August 1998.
- [54] A. Kumar and D. Manjunath. *Communication Networking—An analytical Approach*. Morgan Kaufmann Publishers, Elsevier Inc, San Francisco, USA, 2004.
- [55] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, 1997.
- [56] C. Liu and E. Modiano. On the performance of TCP congestion control over satellite links with ARQ. *Computer Networks*, 47:661–678, 2005.
- [57] S. Low. A duality model of TCP and queue management algorithms. In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, Monterey, CA, USA, September 2000.
- [58] S. Low and D. E. Lapsley. Optimization flow control - I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, 1999.
- [59] S. Low, F. Paganini, and J. C. Doyle. Internet congestion control: An analytical perspective. In *IEEE Control Systems Magazine*, February 2002.
- [60] S. Low, F. Paganini, and L. Wang. Understanding TCP Vegas: A duality model. In *Proceedings of ACM SIGMETRICS*, pages 226–235, Cambridge, Massachusetts, USA, June 2001.
- [61] R. Jain M. Hassan. *High Performance TCP/IP Networking – Concepts, Issues, and Solutions, International Edition*. Pearson Prentice Hall, Pearson Education, Inc. Upper Saddle River, NJ 07458, 2004.
- [62] A. Maheshwari. *Brouwer’s Fixed Point Theorem*.
<http://www.scs.carleton.ca/~maheshwa/MAW/MAW/node3.html>, School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa Ontario, CANADA K1S 5B6, 2003.
- [63] M. A. Marsan, R. Lo Cigno, and M. Meo and E. de Souza e Silva. A Markovian model for TCP over ATM. *Telecommunication Systems*, 12(4):341–368, 1999.
- [64] M. Mathis and J. Mahdavi. Forward acknowledgement: Refining TCP congestion control. In *ACM SIGCOMM*, pages 281–291, Stanford University, California, USA, August 1996.

- [65] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. In *IETF RFC 2018*, October 1996.
- [66] M. Meo, M. Garetto, M. A. Marsan, and R. Lo Cigno. On the use of fixed point approximations to study reliable protocols over congested links. In *Globecom*, San Francisco, USA, December 2003.
- [67] A. Misra, T. Ott, and J. Baras. The window distribution of multiple TCPs with random queues. In *Proceedings of IEEE GLOBECOM*, Rio de Janeiro, Brazil, December 1999.
- [68] J. Mundarath, N. Venkataramaiah, and R. Huang. TCP variants simulation-based analysis: comparison report. Technical report, University of Wisconsin-Madison, Electrical and Computer Engineering, USA, May 2002.
- [69] P. Nain. *Basic Elements of Queueing Theory: Application to the Modeling of Computer Systems, Lecture Notes*. INRIA, 06902 Sophia Antipolis, France, 1998.
- [70] ns2. *Network Simulator (Ver. 2)*. <http://www.isi.edu/nsnam/ns>, ISI, The University of Southern California, 1989.
- [71] Jorgen Olsén. *Stochastic Modeling and Simulation of the TCP Protocol*. PhD thesis, Department of Mathematics, Uppsala University, Box 480, SE-751 06 Uppsala, Sweden, 2003.
- [72] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *ACM SIGCOMM '98*, Vancouver, British Columbia, Canada, August 1998.
- [73] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2):133–145, April 2000.
- [74] F. Paganini. Flow control via pricing: a feedback perspective. In *Proceedings of the Allerton Conference*, Monticello, IL, October 2000.
- [75] F. Paganini, Z. Wang, J. C. Doyle, and S. H. Low. Congestion control for high performance, stability and fairness in general networks. *IEEE/ACM Transactions on Networking*, 13(1):43–56, February 2005.
- [76] L. L. Peterson and B. S. Davie. *Computer Networks – A Systems Approach, 3rd Edition*. The Morgan Kaufmann Series in Networking, Elsevier Science, San Francisco, CA 94104-3205, USA, 2003.

- [77] A. Riedl, T. Bauschert, M. Perske, and A. Probst. Investigation of the M/G/R processor sharing model for dimensioning of IP access networks with elastic traffic. In *Proceedings of First Polish-German Teletraffic Symposium (PGTS)*, ITG im VDE, Dresden, Germany, September 2000.
- [78] J. Roberts and T. Bonald. Performance of bandwidth sharing mechanisms for service differentiation in the Internet. In *Proceedings of 13th ITC Specialist Seminar, IP Traffic Measurement, Modeling and Management*, Monterey, CA, USA, September 2000.
- [79] S. M. Ross. *Introduction to Probability Models, 6th Edition*. Academic Press Limited, San Diego, CA 92101-4495, USA, 1997.
- [80] C. Samios and M. Vernon. Modeling the throughput of TCP Vegas. In *Proceedings of ACM SIGMETRICS*, San Diego, California, U.S.A, June 2003.
- [81] B. Sikdar, S. Kalyanaraman, and K. S. Vastola. An integrated model for the latency and steady-state throughput of TCP connections. *Performance Evaluation*, 46(2):139–154, September 2001.
- [82] B. Sikdar, S. Kalyanaraman, and K. S. Vastola. Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno and SACK. *IEEE/ACM Transactions on Networking*, 11(6):959–971, December 2003.
- [83] W. R. Stevens. *TCP/IP Illustrated*. Addison Wesley Reading, MA, USA, 1994.
- [84] J. Sun, M. Zukerman, K.T Ko, G.Chen, and S.Chan. Effects of large buffers on TCP queueing behavior. In *INFOCOM 2004, Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, Hong Kong, March 2004.
- [85] A. Tang, J. Wang, and S. H. Low. Counter-intuitive throughput behavior in networks under end-to-end control. *IEEE/ACM Transactions on Networking*. to appear 2006.
- [86] A. Tang, J. Wang, S. H. Low, and M. Chiang. Network equilibrium of heterogeneous congestion control protocols. In *IEEE INFOCOM*, Miami, FL USA, March 2005.
- [87] K. A. Tang, J. Wang, S. Hegde, and S. H. Low. Equilibrium and fairness of networks shared by TCP Reno and FAST. *Telecommunications Systems special issue on High Speed Transport Protocols*. to appear 2005.
- [88] K. Thompson, G. J. Miller, and R. Wilder. Wide-area Internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–23, November 1997.

- [89] K. Thompson, G. J. Miller, and R. Wilder. Wide-area Internet traffic patterns and characteristics (extended version). Technical report, MCI Telecommunications Corporation, <http://netlab.cs.tsinghua.edu.cn/~zm/papers/MCItraffic.pdf>, October 2003.
- [90] P. Tinnakornsrisuphap, W. C. Feng, and I. Philp. On the burstiness of the TCP congestion-control mechanism in a distributed computing system. In *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, Taipei, Taiwan, April 2000.
- [91] P. Trimintzios, P. Flegkas, G. Pavlou, L. Georgiadis, and D. Griffin. Policy-based network dimensioning for IP differentiated services networks. In *Proceedings of the IEEE Workshop on IP Operations and Management (IPOM 2002)*, pages 171–176, Dallas, Texas, USA, October 2002.
- [92] R. Vranken, R. D. van der Mei, R. E. Kooij, and J. L. van den Berg. Performance of TCP with multiple priority classes. In *COST 279*, Lyngby, Denmark, September 2002.
- [93] J. Wang, L. Li, S. H. Low, and J. C. Doyle. Cross-layer optimization in TCP/IP networks. *IEEE/ACM Transactions on Networking*, 13(3):568–582, June 2005.
- [94] E. W. Weisstein. *Ball*. From MathWorld—A Wolfram Web Resource, <http://mathworld.wolfram.com/Ball.html>.
- [95] E. W. Weisstein. *Convex Set*. From MathWorld—A Wolfram Web Resource, <http://mathworld.wolfram.com/ConvexSet.html>.
- [96] E. W. Weisstein. *Convolution*. From MathWorld—A Wolfram Web Resource, <http://mathworld.wolfram.com/Convolution.html>.
- [97] E. W. Weisstein. *Geometric Distribution*. From MathWorld—A Wolfram Web Resource, <http://mathworld.wolfram.com/GeometricDistribution.htm>.
- [98] E. W. Weisstein. *Hazard Function*. From MathWorld—A Wolfram Web Resource, <http://mathworld.wolfram.com/HazardFunction.html>.
- [99] E. W. Weisstein. *Newtons Method*. From MathWorld—A Wolfram Web Resource, <http://mathworld.wolfram.com/NewtonsMethod.html>.
- [100] A. Wierman, T. Osogami, and J. Olsén. Modeling TCP-Vegas under On/Off traffic. In *Proceedings of Fifth Workshop on MATHematical performance Modeling and Analysis (MAMA)*, San Diego, California, USA., June 2003.

- [101] A. Wierman, T. Osogami, and J. Olsén. A unified framework for modeling TCP Vegas, TCP SACK, and TCP Reno. In *The 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'03)*, Orlando, USA, October 2003.
- [102] W. L. Winston. *Operations Research: Applications and Algorithms, 3rd Edition*. Wadsworth Inc. belmont, California 94002, USA, 1994.
- [103] I. Yeom and A. L. Narasimha Reddy. Modeling TCP behavior in a differentiated services network. *IEEE/ACM Transactions on Networking*, 9(1):31–46, February 2001.

